KNOWLEDGE BASE REFINEMENT AND KNOWLEDGE TRANSLATION WITH MARKOV LOGIC NETWORKS

by

SHANGPU JIANG

A DISSERTATION

Presented to the Department of Computer and Information Science
and the Graduate School of the University of Oregon
in partial fulfillment of the requirements
for the degree of
Doctor of Philosophy

December 2015

DISSERTATION APPROVAL PAGE

Student: Shangpu Jiang

Title: Knowledge Base Refinement and Knowledge Translation with Markov Logic Networks

This dissertation has been accepted and approved in partial fulfillment of the requirements for the Doctor of Philosophy degree in the Department of Computer and Information Science by:

Dr. Dejing Dou	Chair
Dr. Daniel Lowd	Core Member
Dr. Andrzej Proskurowski	Core Member
Dr. Reza Rejaie	Core Member
Dr. Jonathan Brundan	Institutional Representative

and

Dr. Scott L. Pratt	Dean of the Graduate School

Original approval signatures are on file with the University of Oregon Graduate School.

Degree awarded December 2015

DISSERTATION ABSTRACT

Shangpu Jiang

Doctor of Philosophy

Department of Computer and Information Science

December 2015

Title: Knowledge Base Refinement and Knowledge Translation with Markov Logic Networks

Machine learning and data mining have provided plenty of tools for extracting knowledge from data. Yet, such knowledge may not be directly applicable to target applications and might need further manipulation: The knowledge might contain too much noise, or the target application may use a different representation or terminology.

In this dissertation, we study three problems related to knowledge management and manipulation. First, given a knowledge base (KB) automatically extracted from the text, we explore how to refine it based on the dependencies among the possible KB instances and their confidence values. Second, when the target application to which we want to apply our knowledge uses a different schema, we explore how to translate the knowledge based on the mapping between the schemas. Sometimes, the mapping between two schemas can be discovered automatically, so the third problem we consider is whether we can find the mapping more accurately using the corresponding knowledge contained in the two schemas.

We notice that a large fraction of data and knowledge can be represented in relational models, which can be formalized with first-order logic. Moreover, uncertainty is a common feature existing in these problems, e.g., the confidence values associated with the KB instances, the probabilistic knowledge rules to be translated, or the schemas not perfectly aligned with each other. Therefore, we adopt statistical relational learning, which combines first-order logic with probabilistic models, to resolve these problems. In particular, we use Markov logic networks (MLNs), which consist of sets of weighted first-order formulas. MLNs are a powerful and flexible language for representing hard and soft constraints of relational domains.

We develop the MLN formulations for each of these problems, and we use the representation, inference and learning approaches in the literature with certain adaptations to solve them. The experiment results show that MLNs successfully provide solutions to these problems or achieve better performances than the existing methods.

This dissertation includes previously published and unpublished coauthored material.

CURRICULUM VITAE

NAME OF AUTHOR: Shangpu Jiang

GRADUATE AND UNDERGRADUATE SCHOOLS ATTENDED:

University of Oregon, Eugene, OR, USA
Tsinghua University, Beijing, China

DEGREES AWARDED:

Doctor of Philosophy, Computer and Information Science, 2015, University of Oregon
Master of Science, Computer Science and Technology, 2010, Tsinghua University
Bachelor of Science, Computer Science and Technology, 2007, Tsinghua University

AREAS OF SPECIAL INTEREST:

Machine learning, data mining

PROFESSIONAL EXPERIENCE:

Graduate Research & Teaching Assistant, Department of Computer and Information Science, University of Oregon, 2010 to present

Research Intern, SRI International, Menlo Park, California, 2013

GRANTS, AWARDS AND HONORS:

Graduate Teaching & Research Fellowship, Computer and Information Science, 2010 to present

NSF Travel Grant, International Conference on Data Mining, 2012

PUBLICATIONS:

Jiang, S., Lowd, D., Kafle, S., and Dou, D. (2015). Ontology Matching with Knowledge Rules. Submitted to *Journal on Large-Scale Data and Knowledge-Centered Systems*.

Jiang, S., Lowd, D., and Dou, D. (2016). A Probabilistic Approach to Knowledge Translation. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)*. (To appear).

Jiang, S., Lowd, D., and Dou, D. (2015). Ontology Matching with Knowledge Rules. In *Proceedings of the 26th International Conference on Database and Expert Systems Applications (DEXA 2015)*, Part I, Pages 94-108. (Best Paper Award)

Jiang, S., Lowd, D., and Dou, D. (2012). Learning to Refine an Automatically Extracted Knowledge Base Using Markov Logic. In *Proceedings of the 12th IEEE International Conference on Data Mining (ICDM 2012)*, Pages 912-917.

ACKNOWLEDGEMENTS

I am so grateful to have two enthusiastic, knowledgable and supportive advisors, Dr. Dejing Dou and Dr. Daniel Lowd. Over the five years, they have patiently taught me to develop the essential skills for a PhD student, from identifying and defining an important and influential problem, to writing a research paper. This work would not have been possible without their guidence and many insightful discussions with them.

I would like to thank the National Science Foundation for funding this research through award IIS-1118050 with Dejing Dou as PI and Daniel Lowd as Co-PI, and thank the department of CIS for providing financial and other support. I would also like to thank other members of my dissertation committee for the comments and suggestions.

Finally, thank you to my family, and especially to my wife Xuemei Wan, for their unconditional support.

To my wife, Xuemei, and my daughter, Yuchen

TABLE OF CONTENTS

Chapter Page

Chapter Page

LIST OF FIGURES

Figure Page

LIST OF TABLES

Table Page

CHAPTER I

INTRODUCTION

Today, the amount of *data* is growing at a dramatic speed with the advances of the Internet, social networks and sensors built into mobile devices. Yet people thirst for *knowledge*, abstracted from data, which is a valuable resource for many applications and tasks of our daily lives. Fortunately, with the development of machine learning and data mining, we now have plenty of tools for discovering knowledge from data automatically. This knowledge may have various forms: deterministic or probabilistic, predictive or descriptive. In order to be used in computer systems, it is often represented in formal languages. For example, we may learn the fact that the `Lakers` are a `Basketball` team from a sentence on a web page, which is represented as a first-order atomic formula $\texttt{TeamPlaysSports}(\texttt{Lakers}, \texttt{Basketball})$, or, we may learn a predictive rule that a person who is older than 40 and who works in a university has good financial credit, which is represented as

$$\texttt{Age}(x) \wedge \texttt{Employer}(x, y) \wedge \texttt{Type}(y) = \texttt{university} \Rightarrow \texttt{credit}(x) = \texttt{good}.$$

Usually, this kind of knowledge can be directly used to answer queries or make predictions in certain tasks. Sometimes, however, the knowledge acquired from data is not applicable immediately and requires additional postprocess. In this dissertation, we consider two scenarios, and we propose solutions to the problems arising from them.

In the first scenario, we have a *knowledge base* (KB) containing statements that are automatically extracted from web pages or other sources, such as `TeamPlaysSports`(`Lakers`, `Basketball`) or `Athlete`(`Tiger Woods`) representing `Tiger Woods` is an `Athlete`. Such *information extraction* (IE) systems would often include incorrect statements and ignore correct statements implied in the text. If we could identify contraditions of the facts in the KB, we would potentially be able to remove the wrong facts and add the correct ones. So the first question is *how to refine an automatically extracted knowledge base*.

In the second scenario, we have learned knowledge from a database, but the task to which we want to apply the knowledge uses a slightly different schema (a schema is the terminology and structure used to organize the data and knowledge). We want to convert the knowledge in the source schema to another form in the target schema based on the mapping between them. So the second question is *how to translate knowledge from one schema to another given the mapping between the schemas*. Sometimes, the mapping of schemas is manually created by domain experts. Alternatively, the mapping can be automatically discovered based on the similar names and structures of the two schemas, which is a process called *ontology matching*. In fact, if we also know the corresponding knowledge represented in the two schemas, we can often find a more correct mapping. We name this process *knowledge aware ontology matching* (KAOM). So the third question, which is related to the second one, is *how to discover the mapping between two schemas by exploiting knowledge in addition to other information*.

Our solution to all three of these problems is statistical relational learning (SRL) (Getoor and Taskar, 2007). SRL is a subarea of machine learning that is concerned with probabilistic frameworks for relational (i.e., first-order) domains.

A relational domain is a powerful model for a large fraction of data, knowledge and knowledge bases in our daily applications, and first-order logic (FOL) is the formal language for a relational domain. Yet, uncertainty is often presented in the knowledge. For example, a knowledge base instance extracted from the text may be uncertain; domain rules crafted by experts may be uncertain; the mapping between two schemas may also be uncertain. SRL combines the power of FOL in representing complex structured information and the power of probabilistic models in representing uncertainty, and therefore becomes a perfect choice for tasks related to knowledge manipulation. In particular, we use *Markov logic networks* (MLNs) (Domingos and Lowd, 2009), the most powerful SRL models to date. We adopt representation, inference and learning methods for MLNs proposed in the literature, and we engage certain adaptations and improvements to make them work for the problems of our interest. We also conduct thorough experiments relating all three tasks to verify the effectiveness of our methods.

1.1. Dissertation Outline

The remainder of the dissertation is organized as follows. In Chapter II, we introduce the representations of data and knowledge, as well as the basics of Markov logic networks. In Chapters III, IV and V, we discuss knowledge base refinement, knowledge aware ontology matching and knowledge translation respectively. In Chapter VI, we conclude with future work.

Chapters III, IV and V contain previously published and unpublished coauthored material.

CHAPTER II

BACKGROUND

In this chapter, we will introduce several pieces of background concepts and previous work. First, a *data model* is a formalization of real world data and knowledge for computer systems to manage and manipulate them, which is the foundation of the topics in the dissertation. Ontologies and relational databases are two widely used data models based on first-order logic. Second,the *mapping* between schemas of two data models is utilized in the knowledge translation task, and is also the output of the ontology matching task. Third, *statistical relational learning (SRL)* extends first-order logic with uncertainty, which is a perfect choice for representation, inference, and learning tasks for relational domains and relational knowledge. Last, *Markov logic networks*, the most powerful SRL models to date, are of particular interest in solving the problems proposed in this dissertation and introduced in detail.

2.1. Data Models: Ontologies and Relational Databases

In computer science, a data model is a formal definition and representation of concepts, entities, and their attributes and relationships for a domain of discourse. Ontologies and relational databases are two mainstream data models with rich structural information invented and developed in the AI and database communities respectively.

As formal representations, ontologies and relational databases are both equipped with logic systems for the purpose of reasoning and querying. These logic

systems are all subsets of *first-order logic* (FOL), and as a result, the domains they represent are called relational (i.e., first-order) domains.

2.1.1. First-Order Logic

The syntax of first-order logic theory is defined by a *signature* $\Sigma = (\mathcal{F}, \mathcal{P}, r)$, where $\mathcal{F}$ is a set of *function* symbols, $\mathcal{P}$ is a set of *predicate* symbols, and $r : \mathcal{F} \cup \mathcal{P} \to \mathbb{N}$ maps the function and predicate symbols to their arities. A *term* is either a *variable* symbol (e.g., x) or a function f applied on $r(f)$ terms. In particular, a 0-ary function symbol is a *constant*. An *atomic formula* or *atom* is a predicate P applied on $r(P)$ terms. A *well-formed formula* (wff) is either an atomic formula, or *conjunction* $\wedge$, *disjunction* $\vee$, *negation* $\neg$, *implication* $\to$ of wffs, or $\forall x\phi$ or $\exists x\phi$ where ϕ is a wff containing x and x is said to be *bounded*. A *literal* is an atomic formula or its negation, and a *clause* is a disjunction of literals. A *sentence* ϕ is a wff without free variables. A logical *theory* Γ is a set of sentences.

The semantics of FOLs is defined through an *interpretation* $\mathcal{I} = (\mathcal{D}, \cdot^{\mathcal{I}})$ over a signature Σ, where $\mathcal{D}$ is the *domain* or *universe*, i.e., the set of objects in the domain of discourse, and $\cdot^{\mathcal{I}}$ is a mapping from function and predicate symbols to actual functions and relations over the domain. Specifically, $f^{\mathcal{I}} \in \{\tilde{f}, \tilde{f} : \mathcal{D}^{r(f)} \to \mathcal{D}\}$, $P^{\mathcal{I}} \in \{\tilde{P}, \tilde{P} \subseteq \mathcal{D}^{r(P)}\}$. The evaluation of a sentence ϕ with respect to an interpretation $\mathcal{I}$ is recursively defined, where the connectives are defined the same as propositional logic, and the quantifiers $\forall x\phi$ and $\exists x\phi$ are defined based on $\phi[x\backslash c]$, the result of substituting $c \in \mathcal{D}$ for every free occurrence of x in ϕ. An interpretation $\mathcal{I}$ is said to be a *model* of a theory Γ, denoted $\mathcal{I} \models \Gamma$, when all sentences in Γ are true. A theory Γ *entails* a sentence ϕ, denoted $\Gamma \models \phi$, when every model of Γ is also a model of ϕ.

In practise, we use meaningful phrases as predicate and function symbols. For example, `Basketball` and `Tiger Woods` are constants, and `TeamPlaysSport`(t, s) is a predicate that is true if team t plays sport s. In this dissertation, we do not consider functions except for constants. By convention, we use lowercase identifiers for variables and capitalized identifiers for constants and predicates. An atomic formula is the application of a predicate to a tuple of variables and/or constants, e.g., `TeamPlaysSport`$(t,$ `Basketball`$)$. A ground formula or ground atom is a formula or an atom where all variables have been substituted by constants.

2.1.2. Ontologies

First-order logic is undecidable. Therefore, decidable subsets of FOL are usually used for effective reasoning. In ontologies, the family of description logic (DL) has been widely used as the logical formalism of ontologies, and is deployed in the Semantic Web and the web language OWL [1].

The components of description logic (and ontologies) are *concepts* (or *classes*), *roles* (or *properties*), *individuals*, and *axioms* describing their relationships. Intuitively, there is a correspondence between DL and a restricted fragment of FOL: A concept corresponds to a monadic predicate in FOL, a role corresponds to a dyadic predicate, an individual corresponds to a constant, and an axiom corresponds to an FOL sentence.

The DL axioms are divided into the *terminological box* (TBox), which contains no individuals (e.g., the inclusion relation of two concepts/roles), and the *assertion box* (ABox), which contains individuals (e.g., the concept/role instances). The axioms may contain concepts and roles that are compositions of basic concepts

[1] `http://www.w3.org/TR/owl2-primer/`

and roles, which are also called complex classes and properties in OWL. See examples of TBox and ABox axioms and constructors in Table 2.1.

TABLE 2.1. (From top to bottom) Syntax and semantics of common DL symbols, constructors, TBox assertions, and ABox assertations

Description	Syntax	Semantics
Everything	$\top$	$\mathcal{D}$
Empty	$\bot$	$\emptyset$
Concept	C	$C^{\mathcal{I}} \subseteq \mathcal{D}$
Role	R	$R^{\mathcal{I}} \subseteq \mathcal{D} \times \mathcal{D}$
Individual	a	$a^{\mathcal{I}} \in \mathcal{D}$
Intersection	$C \sqcap D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}}$
Union	$C \sqcup D$	$C^{\mathcal{I}} \cup D^{\mathcal{I}}$
Complement	$\neg C$	$\mathcal{D} \backslash C^{\mathcal{I}}$
Universal restriction	$\forall R.C$	$\{x \in \mathcal{D} \mid \forall y((x,y) \in R^{\mathcal{I}} \rightarrow y \in C^{\mathcal{I}})\}$
Existential restriction	$\exists R.C$	$\{x \in \mathcal{D} \mid \exists y((x,y) \in R^{\mathcal{I}} \wedge y \in C^{\mathcal{I}})\}$
Role composition	$R \circ S$	$\{(x,y) \mid \exists z((x,z) \in R^{\mathcal{I}} \wedge (z,y) \in S^{\mathcal{I}})\}$
Inverse	R^{-}	$\{(x,y) \mid (y,x) \in R^{\mathcal{I}}\}$
Domain restriction	$R \upharpoonright C$	$\{(x,y) \in R^{\mathcal{I}} \mid x \in C^{\mathcal{I}}\}$
Range restriction	$R \downharpoonright C$	$\{(x,y) \in R^{\mathcal{I}} \mid y \in C^{\mathcal{I}}\}$
Subsumption	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
Equivalence	$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$
Disjointness	$C \sqsubseteq \neg D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$
Concept instance	$a : C$	$a^{\mathcal{I}} \in C^{\mathcal{I}}$
Role instance	$(a,b) : R$	$(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

An ontology, in a narrow sense, contains only TBox statements, the conceptualization of a domain. In a broader sense, it contains both TBox and ABox statements, which is also called a *knowledge base*.

2.1.3. Relational Database

Databases, and in particular, relational databases, are the most widely used mechanism for efficiently storing, manipulating, and retrieving data. A relational database is a set of relational tables describing the attributes of objects and the relationships among them.

A relational database is usually designed with an entity-relationship (ER) model, and then normalized to a relational model that can be utilized by a database management system (DBMS). Formally, in a relational model, we have a set of *types* or *attributes* $\mathcal{A}$ and a set of *relations* $\mathcal{R}$. Each relation R_i is a set of *tuples*, where each tuple is a partial function from attribute names to values.

A relational model may also incorporate *integrity constraints* as semantics. The constraints can essentially be arbitrary first-order logic sentences, but are usually restricted subsets of FOL called *dependencies* due to inference feasibility. Similarly, query languages for relational models are usually restricted subsets as well, in particular, the language of *conjuctive queries* and its variants.

2.2. Mapping of Schemas

For the "knowledge aware ontology matching" and "knowledge translation" problems, we introduce previous work in representing the mapping between two schemas of the same domain. This topic is investigated in the database and AI communities, respectively, with different flavors. In the database literature, people focus on how to apply schema mappings for the task of semantic integration, whereas in the AI literature, people focus on developing automatic tools for discovering the alignment of two ontologies (i.e., ontology matching, matching as a verb). Despite of the differences, both types of mappings are essentially subsets of FOL.

2.2.1. Database Schema Mapping

Definition 2.1 (Schema mapping (Lenzerini, 2002))**.** Let $\mathcal{S} = \{S_1, \cdots, S_n\}$ and $\mathcal{T} = \{T_1, \cdots, T_m\}$ be the source and target schemas, respectively. A schema

mapping is a triple $\mathcal{M} = (\mathcal{S}, \mathcal{T}, \sigma)$ where σ is a set of assertions of the forms

$$q_{\mathcal{S}} \rightsquigarrow q_{\mathcal{T}}$$

$$q_{\mathcal{T}} \rightsquigarrow q_{\mathcal{S}}$$

and $q_{\mathcal{S}}$ and $q_{\mathcal{T}}$ are two queries (views) in $\mathcal{S}$ and $\mathcal{T}$ respectively. Here, $\rightsquigarrow$ may have different logical semantics: $\rightarrow$, $\leftarrow$ or $\leftrightarrow$, which correspond to *sound*, *complete* and *exact* mappings, respectively.

This type of schema mapping is called a GLAV (global and local as view) mapping. Other restricted forms include LAV (local as view) and GAV (global as view). Schema mapping plays a central role in semantic integration, including data integration and data exchange. For different types of queries and mappings, the decidability or complexity of semantic integration tasks may vary.

2.2.2. Ontology Matching

Definition 2.2 (Ontology Matching (Euzenat and Shvaiko, 2007))**.** Given two ontologies O_1 and O_2, a *correspondence* is a 3-tuple $\langle e_1, e_2, r \rangle$ where e_1 and e_2 are entities (i.e., classes or properties) of the first and second ontologies respectively, and r is a semantic relation such as equivalence ($\equiv$) and subsumptions ($\sqsubseteq$ or $\sqsupseteq$). An *alignment* is a set of correspondences. *Ontology matching* is the task or process of identifying the correct semantic alignment between the two ontologies. In most cases, ontology matching focuses on equivalence relationships only.

The word "matching" in ontology matching implies an alignment with only one-to-one correspondences. However, more sophisticated ontology matching

systems can also discover complex correspondences, which are essentially many-to-many correspondences.

Definition 2.3 (Complex Correspondences)**.** A *complex concept* is a composition (e.g., unions, complements) of one or more simple classes or properties. [2] In OWL, there are several constructors for creating complex classes and properties (See Table 2.1). A *complex correspondence* is an equivalence relation between a simple class or property and a complex class or property in two ontologies (Ritze et al., 2008).

2.3. Statistical Relational Learning

One key weakness of a deterministic language such as first-order logic is that it is very brittle: A single inconsistency renders the entire model false. In the real world, knowledge is often uncertain. Even with perfect knowledge of the world, many events are inherently stochastic. For example, in a social network, a person who has a smoking friend is *likely* to smoke as well. This cannot be simply represented with a first-order logic formula

$$\forall x, y \; \texttt{Friend}(x, y) \wedge \texttt{Smoke}(x) \Rightarrow \texttt{Smoke}(y).$$

The area of statistical relational learning (SRL) (Getoor and Taskar, 2007) explores representation, learning, and inference of *probabilistic models* in relational domains. For instance, with Markov logic networks (MLNs), one type of SRL model, we can attach a *weight* to the above formula to indicate the degree of certainty of a formula.

[2] Although the name is complex concept, it actually refers to both complex classes and complex properties, such as unions or role compositions.

Relational domains are powerful enough for many daily scenarios and tasks. In traditional machine learning and data mining, a less sophisticated setting is usually considered: The datasets consist of a set of independent and identically distributed (i.i.d.) data instances, each of them being a fixed length vector of attributes. This type of data is called *propositional data*, and can be stored in a single table in the *attribute-value representation*. A traditional (propositional) data mining process can discover knowledge patterns from the propositional data and use the knowledge to deal with various tasks, such as classification, regression, clustering, and anomaly detection. Therefore, SRL can also be considered as an extension of traditional probabilistic methods to relational domains.

2.3.1. Knowledge

The term *knowledge* generally refers to information we gained from experience or data. In the context of "knowledge aware ontology matching" and "knowledge translation", knowledge is defined (informally) as statements about classes, attributes or relations of a relational domain, which we would like to distinguish from *data*. Here, data refers to data instances, which are facts about individuals, whereas knowledge refers to meta-theorems about data instances. For example, a TBox axiom is considered as knowledge, and an ABox axiom is considered as a data instance.

Since SRL extends FOL and DL with uncertainty, it seems to be the most powerful method for representing knowledge in a relational domain.

2.4. Markov Logic Networks

A Markov logic network (MLN) (Domingos and Lowd, 2009) or Markov logic consists of a set of weighted formulas in first-order logic, $\{(F_i, w_i)\}$. For example, in a social network, we define a set of first-order predicates (e.g., `Friends`, `Smokes`, `Cancer`) to represent the attributes and relationships. A Markov logic in this domain could be

$$0.7 \quad \texttt{Friend}(x, y) \wedge \texttt{Friend}(y, z) \Rightarrow \texttt{Friend}(x, z)$$

$$1.5 \quad \texttt{Smokes}(x) \Rightarrow \texttt{Cancer}(x)$$

$$0.5 \quad \texttt{Friend}(x, y) \wedge \texttt{Smoke}(x) \Rightarrow \texttt{Smoke}(y)$$

Here and for the rest of the dissertation, formula weights are shown to the left of the formula, and hard formulas are represented by placing a period (.) at the end of the formula. Intuitively, each formula F_i represents a noisy first-order rule, and its weight w_i indicates the relative strength or importance of that rule.

Together with a *finite* set of constants (i.e., the *domain*), an MLN defines a probability distribution over possible worlds or Herbrand interpretations (the truth value assignment to all ground atoms) by:

$$p(\boldsymbol{X} = \boldsymbol{x}) = \frac{1}{Z} \exp\left(\sum_i w_i n_i(\boldsymbol{x})\right)$$

where $n_i(\boldsymbol{x})$ is the number of satisifed groundings of F_i in the possible world $\boldsymbol{x}$ and Z is a normalization constant. In the above example, we define a domain containing two constants `Anna` and `Bob`. The Herbrand base (i.e., the set of all ground atoms) would be `Friends(Anna, Bob)`, `Friends(Bob, Anna)`, `Friends(Anna,`

`Anna`), `Friends`(`Bob`, `Bob`), `Smokes(Anna)`, `Smokes(Bob)`, `Cancer(Anna)`, `Cancer(Bob)`.

An MLN is a *log-linear model* of a relational domain. A log-linear model is a compact way to represent a probability distribution $p(\boldsymbol{X})$ over a set of random variables $\boldsymbol{X} = \{X_1, X_2, \ldots, X_N\}$:

$$p(\boldsymbol{X} = \boldsymbol{x}) = \frac{1}{Z} \exp\left(\theta^T \phi(\boldsymbol{x})\right),$$

where $\phi(\boldsymbol{x})$ is a vector of feature functions, θ is a vector of weights, and Z is a normalization constant. For an MLN, the feature functions are $n_i(\boldsymbol{x})$, the number of satisfied ground formula of each formula F_i.

In a propositional domain, many popular probabilistic graphical models, such as Bayesian networks (Pearl, 1988) and Markov random fields (MRFs (Kindermann and Snell, 1980)), can be represented as log-linear models. Markov logic networks can be considered as an extension of Markov random fields into relational domains.

2.4.1. Inference

Inference helps us reason probabilistically about complex relations in Markov logic networks. There are two basic types of inference: most probable explanation (MPE) or maximum a-posterior (MAP) inference that finds the most probable state of the world consistent with some evidence, as well as probabilistic inference that finds the conditional/marginal distribution of a formula or a predicate. Theorectically, any inference algorithm for Markov random field can be applied to MLNs, but specialized algorithms that exploit the structure of MLNs often give better performance.

The MPE inference task can be reformulated as maximizing the total weight of satisfied ground formulas, and thus solved with MaxWalkSAT (Domingos and Lowd, 2009). MaxWalkSAT iteratively searches for the highest-weight possible world using a random walk. In each iteration, MaxWalkSAT picks a random unsatisfied clause and changes the truth assignment of one of the atoms in the clause. With probability p, it selects the "best" atom that greedily maximizes the resulting sum of satisfied formula weights, and with probability $1 - p$ it selects a random atom. The mixed strategy allows the algorithm to prefer better configurations without getting stuck in local optima. Alternatively, MPE inference can also be reformulated as integer linear programming (ILP), and solved with its linear programming (LP) relaxation (Riedel, 2008).

Standard probabilistic inference methods for Markov random fields such as Gibbs sampling and belief propagation often fail in MLN because of the presence of deterministic or nearly-deterministic formulas. MC-SAT (Poon and Domingos, 2006) is a Markov chain Monte Carlo (MCMC) algorithm that adopts a SAT solver to jump between isolated modes of a distribution with a mixture of hard and soft constraints which are common in MLNs. Other sampling methods that handle hard and soft constraints include SampleSearch (Gogate and Dechter, 2011) and GiSS (Venugopal and Gogate, 2013).

2.4.2. Scalability

Inference algorithms need to first instantiate the FOL formulas into propositional ones, by subsituting the universally quantified logical variables with constants in a domain. This requires memory on the order of the number of constants raised to the number of logical variables in the clauses, which is typically

extremely large. Lazy inference (Singla and Domingos, 2006; Poon et al., 2008) takes advantage of the sparseness in typical relational domains (most ground predicates are false, and most clauses are trivially satisfied), by only putting into memory the non-default value ground predicates and clauses. With lazy inference, the memory cost does not scale with total number of groundings, but only with the number of non-default value groundings. Lazy variants have been developed for both MaxWalkSAT and MC-SAT.

For linear programming solvers, the cutting-plain method can be used to avoid including all the constraints at the beginning, but instead adding them as needed (Riedel, 2008).

2.4.3. Weight Learning

Supervised weight learning of Markov logic minimizes the negative conditional likelihood

$$\mathcal{L}(w|x,y) = -\log P_w(Y = y|X = x)$$

where X is the evidence and Y is the query predicates. The gradient is calculated by

$$\frac{\partial \mathcal{L}(w|x,y)}{\partial w_i} = E_{w,y}[n_i(x,y)] - n_i(x,y)$$

and Hessian by

$$\frac{\partial^2}{\partial w_i \partial w_j}\mathcal{L}(w|x,y) = E_{w,y}[n_i n_j] - E_{w,y}[n_i]E_{w,y}[n_j]$$

where w_i is the weight of the ith formula, n_i is number of truth groundings for the ith formula, and all the terms are conditioned on x. Lowd and Domingos (2007a) discuss several first-order and second-order gradient descent optimization methods,

and found that scaled conjugate gradient with preconditioner (PSCG) performed well in several datasets.

2.4.4. Applications

MLNs have been used for joint inference in information extraction. Poon and Domingos (2007) used Markov logic, the MC-SAT algorithm, and an integrated inference process to extract and match database records from CiteSeer text datasets. With a joint inference, the segmentation of all records and entity resolution are performed together. Their approach consistently improves accuracy over non-joint inference. One of the strengths of MLNs is in performing joint inference over a set of related, uncertain facts. For example, Singla and Domingos (2006) perform entity resolution based on Markov logic by jointly inferring which pairs of bibliographic entries refer to the same paper. Their approach simultaneously infers equivalences among paper authors, titles, and venues, which allows a small number of formulas in Markov logic to capture the essential features of many different approaches to the entity resolution problem.

CHAPTER III

REFINING A KNOWLEDGE BASE WITH MARKOV LOGIC NETWORKS

This work was published in the Proceedings of the 12th IEEE International Conference on Data Mining (ICDM 2012). I was the primary contributor to the methodology and writing, and designed and conducted the experiments. The co-authors contributed partly to the methodology and writing. Dejing Dou and Daniel Lowd were the principle investigators for this work.

3.1. Introduction

The objective of information extraction (IE) is to extract information from natural language text into machine understandable *knowledge base* (KB). Early IE systems focus on a specific domain and certain types of information. For example, an IE system might recognize and extract information regarding business acquisitions and mergers from a set of newspaper articles, while ignoring other types of information. The emergence of the World Wide Web provides a great oppertunity for IE systems to leverage billions of web pages and a vast amount of unstructured or semi-structured information, in the form of natural language, on these web pages. Since then, many *web-scale*, *domain-independent* IE systems have been proposed, such as WebKB (Craven et al., 1999), KnowItAll (Etzioni et al., 2005), TextRunner (Banko et al., 2007), and NELL (Carlson et al., 2010a). Often, the output of these systems are pairs $t = (e, c)$ representing that an entity e is of class c, and triples $t = (s, r, o)$ representing a relation r between two entities s and o.

In *open IE* systems such as KnowItAll and TextRunner, the relations are not defined in advance, and the extraction rules are relation-independent ones that can identify new relations directly from the texts. In other systems such as NELL, a finite set of classes and relations is defined in advance, and an *ontology* is sometimes used to guide the extraction process and organize the extracted data instances. Such ontology-based IE (OBIE (Wimalasuriya and Dou, 2010)) systems thus populate an ontology-based knowledge base as output.

Due to the size of the web, many of these systems adopt unsupervised or semi-supervised learning paradigms. For example, both TextRunner and NELL use bootstrapping approaches, which start with some seed instances (e.g., `France` is a instance for `country`), learn extractors with them (e.g., "countries such as _"), and use these extractors to generate more instances, and so on and so forth. Such a bootstrapping paradigm is very sensitive to errors, which are inevitable for automatic extraction. Once an error occurs, it would propagate and cause more errors.

If an IE system is equipped with an ontology, the ontology can be utilized to improve the quality of the knowledge base by identifying errors that violate the ontological constraints. In NELL, at each iteration of its bootstrapping learning process, only a few candidate instances can be promoted to the KB. The candidates are not only ranked by their confidence scores from the extractors, but also filtered for whether they violate the ontological constraints and existing instances in the KB. The filtering process is called coupled learning (Carlson et al., 2010b). A problem of coupled learning is that it always assumes the existing KB instances are correct when a contradiction occurs. When NELL incorporates incorrect instances in its knowledge base, these instances could lead it to exclude correct but

contradictory instances from being added later on, even if the new instances were supported by overwhelming evidence. NELL also ignores the relationship between the uncertainty of different candidate instances. If multiple related instances have a modest amount of support, then they all should be likely to be true. On the other hand, if a contradictory instance has some support, that should decrease the probability that a given instance is true.

Statistical relational learning (SRL) techniques such as Markov logic networks combine first-order logic with probabilistic graphical models. Since NELL's knowledge base is highly uncertain, and its ontology defines many relational constraints, statistical relational learning seems like a good fit for information extraction. For example, in related work, Poon and Domingos (2010) simultaneously extracted KB instances and learned an ontology for biomedical text. However, current SRL techniques, including Markov logic, do not yet reliably work at web scale.

In order to handle both the large scale and uncertainty in the web, we present a new method for automatically cleaning a noisy knowledge base using Markov Logic Network. Our method performs joint probabilistic inference over many candidate instances. Ontological constraints from the original IE system are translated as hard constraints in the MLN, while confidence values on individual instances are translated as soft constraints. We explore several different methods for turning confidence values from the IE system into weights of the MLN formulas. The simplest approach is to let the weights equal the raw confidence values. A more sophisticated approach is to use either the system's beliefs or human-generated labels as training data, such that a "calibrated confidence" can be computed for each instance. We consider both the standard MLN learning

algorithms and an approximate algorithm, logistic regression, for the sake of efficiency.

Our method achieves scalability by working on an extracted knowledge base, rather than the original text corpus, which could contain millions or billions of web pages. Since the extracted knowledge base could still be very large, we introduce a novel neighborhood-based grounding procedure, which selects a tractable subset of the knowledge base to reason about. By rotating through different subsets, we can clean a very large knowledge base without running out of memory. Different subsets can also be run in parallel.

To evaluate this method, we apply several versions of our MLN and grounding procedure to NELL and show that running joint inference usually leads to higher accuracy, as measured by the area under the precision-recall curve (AUC) and F1. Furthermore, we look at examples of specific instances and investigate how joint reasoning helps to predict their correct values.

The rest of the chapter is organized as follows. Section 3.2. gives brief introductions to NELL and other related work. Section 3.3. describes our MLN-based approach in detail. Section 3.4. shows the experiments and analyzes the results. Section 3.5. concludes and discusses some directions for future work.

3.2. Background and Related Work

3.2.1. Never Ending Language Learner

We use the Never-Ending Language Learner (NELL) system (Carlson et al., 2010a; Mitchell et al., 2015) as a case study to explore methods for refining automatically extracted knowledge bases. NELL is an information extraction system proposed and implemented by a group of researchers at Carnegie Mellon

University. The final goal of NELL is to create an AI system that runs 24 hours per day, 7 days per week, forever, performing two tasks each day:

- Reading task: extract information from web and populate a knowledge base containing structured facts.
- Learning task: improve its reading ability so that it can extract more facts from the web, more accurately.

NELL starts from a small number of "seed instances" of each category and relation in the seed ontology. It uses natural language processing and information extraction techniques to extract candidate instances from a large web corpus, using the current facts in the knowledge base as training examples. The four subcomponents that extract candidates are *Pattern Learner*, *SEAL*, *Morphological Classifier*, and *Rule Learner*, where most candidates are extracted from the first two subcomponents. The Pattern Learner is a free-text extractor which learns and uses contextual patterns such as "mayor of X" and "X plays for Y" to extract instances of categories and relations. The extraction patterns are learned using the co-occurrence statistics between noun phrases and contextual patterns. SEAL is a semi-structured extractor which queries the webpages with instances, and which mines lists and tables to learn new instances of the corresponding predicate. It is based on the assumption that the entities showing up in the same list or table tend to belong to the same category or have the same relation. The Morphological Classifier uses a set of binary L_2-regularized logistic regression models to classify noun phases based on various morphological features. The Rule Learner uses an algorithm similar to FOIL to learn probabilistic Horn clauses. The learned rules are used to infer new relation instances from the current KB.

After extracting candidates, NELL's Knowledge Integrator (KI) promotes candidate facts to beliefs when they have support from multiple extraction components and/or a very high confidence from a single component. However, candidate category instances are not promoted if they already belong to a mutually exclusive category, and relation instances are not promoted unless their arguments are at least candidates for the appropriate category types. NELL heuristically promotes the most likely instances, updates its information extraction systems, and repeats the process, continually expanding its knowledge base and refining its extraction sub-systems. This *bootstrap learning* method takes advantage of the tremendous redundancy in the web corpus. It does not need perfect extraction rules, because multiple pieces of evidence for a new instance can be used to support its correctness.

A major problem of NELL is that the accuracy of the knowledge it acquires gradually decreases as it continues to operate. After the first month, NELL had an estimated precision of 0.9; after two more months, precision had fallen to 0.71, nearly tripling the fraction of incorrect extractions. The underlying reason is that the extraction patterns are not perfectly reliable, so some false instances are extracted as well. The false instances will be used to extract more and more unreliable extraction patterns and false instances, eventually dominating the knowledge base. This kind of error propagation is a common problem of bootstrap learning systems. As discussed earlier, NELL uses ontological constraints to filter out contradictory facts, a method they refer to as coupled training (Carlson et al., 2009, 2010b). Coupled training slows the degradation, but does not entirely prevent it. NELL also uses periodic human supervision to remove incorrect facts. However, human labels are often expensive to obtain.

Recently, Lao et al. (2011) presented an approach to combine constrained, weighted, and random walks through the NELL knowledge base graph to reliably infer new facts for NELL. This approach can learn to infer different target relations by tuning the weights associated with random walks that follow different paths through the knowledge base graph. Like our proposed method, this approach operates on the extracted knowledge base to increases NELL's accuracy. As we discuss later, the two methods are largely orthogonal and could potentially be used together to achieve even higher accuracy.

3.2.2. Probabilistic Soft Logic

Pujara et al. (2013) propose to use probabilistic soft logic (PSL) (Bröcheler et al., 2010) to solve the task of knowledge graph identification (KGI), which is essentially the same task as our task of knowledge base refinement. PSL is a variant of Markov logic which handles continuous truth values in $[0, 1]$ instead of discrete binary values $\{0, 1\}$. In a PSL model with variables $X = \{x_i, i = 1, ..., n\}$, an interpretation is a mapping $I : X \to [0, 1]^n$. The satisfiability of a formula with respect to an interpretation I is defined with Lukasiewicz t-(co)norm:

$$
\begin{aligned}
d(x_1 \tilde{\wedge} x_2; I) &= \max\{0, I(x_1) + I(x_2) - 1\}, \\
d(x_1 \tilde{\vee} x_2; I) &= \min\{I(x_1) + I(x_2), 1\}, \\
d(\tilde{\neg} x_1; I) &= 1 - I(x_1),
\end{aligned}
$$

and the feature of each formula in the log-linear model is defined as

$$
\phi(f) = (1 - d(f; I))^p
$$

where $1 - d(f; I)$ is the unsatisfiability of f, and $p \in \{1, 2\}$ is L_1 or L_2 norm which controls whether completely satisfied formulas would be preferrable (when $p = 1$).

Pujara et al. (2013) use a similar set of formulas as ours, except that each query fact is a continuous random variable instead a binary one. We compare the results of their method with ours in Section 3.4..

3.2.3. Other Related Work

Our research is closely related to ontology-based information extraction (OBIE), which combines information extraction with knowledge representation by using ontologies to guide information extraction (Wimalasuriya and Dou, 2010). Many OBIE systems only extract instances for classes and property values for properties. Such OBIE systems include PANKOW (Cimiano et al., 2004) (as well as its improved version, C-PANKOW (Cimiano et al., 2005)), SOBA (Buitelaar and Siegel, 2006), OntoSyphon (McDowell and Cafarella, 2006), Vulcain (Todirascu et al., 2002), and KIM (Popov et al., 2004). The Kylin system (Wu et al., 2008) constructs an ontology based on the structure of Wikipedia infoboxes. It is interesting to note that constructing an ontology from text and making extractions with respect to that ontology (in the form of individuals and property values) is similar in principle to open information extraction, where relations of interest are automatically discovered from text. Banko et al. (2007) have developed the "TextRunner" IE system, which discovers relations from text using machine learning techniques. In addition, Weld et al. (2008) consider their Kylin system to be an open information extraction system because it discovers relations from infobox classes of Wikipedia, allowing it to discover about 50,000 relations, each having around 10 attributes. Other potentials of OBIE include its ability to create

semantic contents for the Semantic web (Cimiano et al., 2004), and the ability to use it as a mechanism of improving ontologies (Kietz et al., 2000; Maynard, 2006).

3.3. Methodology

In this section, we describe our method in detail. We begin with the Markov logic formulas we use to represent the knowledge base and associated ontology, followed by the extension to incorporate the extraction patterns. We then describe how we make inference and weight learning in this task tractable. We conclude by discussing the extensibility of our method.

3.3.1. Markov Logic Representation of the Knowledge Base and Ontology

The NELL knowledge base has two types of predicates, namely *category* and *relation*. For example, `Athlete`(`Tiger Woods`) means that `Tiger Woods` has the category of `Athlete`, and `TeamPlaysSports` (`Lakers`, `Basketball`) means that `Lakers` are related to `Basketball` by the relation `TeamPlaysSports`. The ontology hierarchy and other constraints can be seen as axioms or rules in first-order logic. For example, we can represent the ontological constraint that every `Athlete` is a `Person` with the rule: `Athlete`$(x) \Rightarrow$ `Person`(x). Similarly, since every bird is an animal, `Bird`$(x) \Rightarrow$ `Animal`(x), and so on.

However, rather than creating predicates in our MLNs for every category and relation in the ontology, we use a more compact representation in which the names of categories and relations (such as `Bird`, `Animal`, etc.) are viewed as constants of type "category" or "relation" in the second-order predicates $\textsc{Cat}(x, c)$ (x is an entity of category c) or $\textsc{Rel}(x, y, r)$ (x and y have relation r).

For example, our sample facts from the previous paragraph would be represented as Cat(`Tiger Words`, `Athlete`) and Rel(`Lakers`, `Basketball`, `TeamPlaysSports`).

In our task, we want to infer the values of $\textsc{Cat}(x, c)$ and $\textsc{Rel}(x, y, r)$. The formulas we use to capture the joint distribution of all the ground predicates are as follows.

3.3.1.1. Ontological constraints

We represent four types of ontological constraints: subsumption among categories and relations (e.g., every bird is an animal); mutually exclusive categories and relations (e.g., no person is a location); inversion (for mirrored relations like `TeamHasPlayer` and `PlaysForTeam`); and the type of the domain and range of each predicate (e.g., the mayor of a city must be a person).

We represent the presence of these constraints using the following predicates: Sub and RSub are the subclass relationships for categories and relations; Mut and RMut are the mutual exclusion relationships for categories and relations; Inv is the inversion; and Dom and Ran are the domain and range relationships.

The MLN formulas to enforce these constraints are as follows:

$$\textsc{Sub}(c1, c2) \wedge \textsc{Cat}(x, c1) \Rightarrow \textsc{Cat}(x, c2).$$

$$\textsc{RSub}(r1, r2) \wedge \textsc{Rel}(x, y, r1) \Rightarrow \textsc{Rel}(x, y, r2).$$

$$\textsc{Mut}(c1, c2) \wedge \textsc{Cat}(x, c1) \Rightarrow \neg\textsc{Cat}(x, c2).$$

$$\textsc{RMut}(r1, r2) \wedge \textsc{Rel}(x, y, r1) \Rightarrow \neg\textsc{Rel}(x, y, r2).$$

$$\textsc{Inv}(r1, r2) \wedge \textsc{Rel}(x, y, r1) \Rightarrow \textsc{Rel}(y, x, r2).$$

$$\textsc{Domain}(r, c) \wedge \textsc{Rel}(x, y, r) \Rightarrow \textsc{Cat}(x, c).$$

$$\textsc{Range}(r, c) \wedge \textsc{Rel}(x, y, r) \Rightarrow \textsc{Cat}(y, c).$$

All of these formulas are maintained as hard constraints, which is equivalent to having an infinitely large weight.

3.3.1.2. Prior confidence of instances

Different facts extracted by an IE system often have different degrees of confidence, based on the amount of supporting evidence available. Rather than simply thresholding or taking the highest-confidence facts consistent with the current knowledge base, Markov logic enables us to reason *jointly* over all facts in order to select an entire set of facts that is mutually consistent and well-supported by evidence.

In our MLN, we use the predicates $\textsc{CandCat}(x, c, conf)$ and $\textsc{CandRel}(x, y, r, conf)$ to represent that x has category c with confidence $conf$, and x and y have relation r with confidence $conf$. The confidences are real numbers provided by the base IE system used to extract the candidate categories and relations. Similarly, we use $\textsc{PromCat}(x, c, conf)$ and $\textsc{PromRel}(x, y, r, conf)$

to represent the instances actually promoted to the knowledge base, with confidence $conf$.

We can incorporate the IE system's confidence by using it as a formula weight for the corresponding ground fact:

$$w_1 \cdot conf \qquad \textsc{CandCat}(x, c, conf) \Rightarrow \textsc{Cat}(x, c) \qquad \text{(Equation 3.1)}$$

$$w_2 \cdot conf \qquad \textsc{CandRel}(x, y, r, conf) \Rightarrow \textsc{Rel}(x, y, r) \qquad \text{(Equation 3.2)}$$

$$w_3 \cdot conf \qquad \textsc{PromCat}(x, c, conf) \Rightarrow \textsc{Cat}(x, c) \qquad \text{(Equation 3.3)}$$

$$w_4 \cdot conf \qquad \textsc{PromRel}(x, y, r, conf) \Rightarrow \textsc{Rel}(x, y, r) \qquad \text{(Equation 3.4)}$$

We can assign a weight to these formulas as well, which would effectively scale all of the confidences by a constant value. In our experiments, we tried uniform weighting, in which we use the original confidences, as well as learning weights using logistic regression. In NELL, confidence values range from 0 to 1, and the promoted confidence for a fact may be different than its candidate confidence.

We also have two formulas for representing the default prior of all the facts:

$$w_5 \quad \textsc{Cat}(x, c) \qquad \text{(Equation 3.5)}$$

$$w_6 \quad \textsc{Rel}(x, y, r) \qquad \text{(Equation 3.6)}$$

3.3.1.3. Seed instances

Seed instances used to initialize the information extraction system are known to be true. We denote these with the SeedCat and SeedRel predicates, for category and relation facts, respectively. For some categories and relations, there may be negative seed examples, which are denoted as NSeedCat and NSeedRel.

We include the seed instances using the following hard formulas:

$$\textsc{SeedCat}(x, c) \Rightarrow \textsc{Cat}(x, c).$$

$$\textsc{NSeedCat}(x, c) \Rightarrow \neg\textsc{Cat}(x, c).$$

$$\textsc{SeedRel}(x, y, r) \Rightarrow \textsc{Rel}(x, y, r).$$

$$\textsc{NSeedRel}(x, y, r) \Rightarrow \neg\textsc{Rel}(x, y, r).$$

3.3.2. Incorporating Extraction Patterns

Many IE systems use extraction patterns or rules as a primary means to generate knowledge (Banko et al., 2007). Extraction patterns are manually or automatically created and may vary considerably in their effectiveness. If we know the reliabilities of extraction patterns used to select candidate facts, then this extra information can help us better determine the truth of candidate facts.

Here we use a simple logistic regression model to predict the truth of candidate facts with the extraction patterns as features.

$$\ln \frac{P(f)}{1 - P(f)} = \beta_0 + \sum_{i=1}^{k} \beta_i \textsc{CoOccur}(f, p_i)$$

where $\textsc{CoOccur}(f, p)$ represents fact f and pattern p cooccur in the text. β_i roughly reflects the reliability of pattern p_i in extracting facts for specific category or relation. Logistic regression outputs the probability $P(f)$ of each candidate fact f being true based on what patterns cooccur with the fact. If the human labels are available, we can use the labels to learn the logistic regression model. When labels are unavailable, we can still use the promoted facts in the knowledge base to as labels for learning the weights of patterns.

Finally the probabilities of candidate facts are incorporated into the Markov logic by these formulas:

$$w_7 \cdot conf \qquad \textsc{PattCat}(x, c, conf) \Rightarrow \textsc{Cat}(x, c) \qquad \text{(Equation 3.7)}$$

$$w_8 \cdot conf \qquad \textsc{PattRel}(x, y, r, conf) \Rightarrow \textsc{Rel}(x, y, r) \qquad \text{(Equation 3.8)}$$

where $\textsc{PattCat}(x, c, conf)$ and $\textsc{PattRel}(x, y, r, conf)$ represent a category fact $\textsc{Cat}(x, c)$ or a relation fact $\textsc{Rel}(x, y, r)$ with probability $conf$ provided by a logistic regression model.

3.3.3. Neighbor-Based Grounding

We used MC-SAT (Poon and Domingos, 2006) to compute the marginal probability of each candidate category and relation fact. However, we needed to modify our inference task in order to make it tractable, as we describe below.

The major problem we face in inference is that the scale of an information extraction system is usually extremely large. For example, NELL extracted more than 943,000 candidate instances by the 165th iteration. These numbers are even larger for the later iterations since the system keeps running and generating more and more candidates. Lazy inference (Poon et al., 2008) is a general approach to reduce complexity for relational inference algorithms. In Markov logic, it assumes that most atoms are false by default and most formulas true, so that it only has to instantiate a few number of necessary atoms and formulas. However, when the whole ground network is densely connected and many atoms are supported by weak evidence, lazy inference still tends to instantiate all those atoms and therefore becomes very inefficient.

We developed an alternate approach for making these particular MLN inference problems tractable. First, we notice that the whole ground network usually forms several clusters, each of which represents a domain. Most connections between atoms are between atoms in the same cluster. Second, for each cluster, we are mainly concerned with the values of the query atoms, which for this task consist of the candidate categories and relations. Other unknown atoms are only useful for their role in correctly inferring the query atoms, and therefore tolerate more error. We treat the query atoms as well as the atoms in the initial unsatisfied clauses as the center of the network. The close neighbors of them are also added in to enable the joint inference, but the distant atoms and formulas are discarded. In practice, we include 2-neighbors of the center atoms. We can safely adopt these two reductions without sacrificing too much accuracy since most discarded groundings are irrelevant to our query.

The idea of this grounding strategy is similar to lazy inference (Poon et al., 2008) or cutting plane inference (Riedel, 2008). Compared to lazy inference, our approach further reduces the complexity for large scale problems by explicitly controlling the size of the grounded network. However, unlike lazy inference, it is not guaranteed to produce the same result, but merely approximates it. Our method is also similar to the expanding frontier belief propagation (EFBP) (Nath and Domingos, 2010). But instead of dynamically calculating a set of atoms affected by new evidence or modified evidence, we generate the set in advance of the inference phase, and thus is more efficient and specific for the task.

3.3.4. Weight Learning of Formulas

In this particular problem, the only formulas that need weights are the soft confidence formulas which relate CandCat, CandRel, PromCat, PromRel, PattCat, and PattRel to the truth of the corresponding category and relation facts (Equation 3.1 to Equation 3.8). So we first consider an approximate method with logistic regression:

$$\ln \frac{P(f)}{1 - P(f)} = \beta_0 + \sum_{i=1}^{k} \beta_i P_i(f)$$

where $P_i(f)$ represents different confidence measures. Each feature also corresponds to a formula in the Markov logic. The weight learning leverages the training labels to automatically determine which measures are best for the specific knowledge base. Since we do not know how a base IE system calculates the confidences, we can add some simple transformations of the original confidences (e.g., log-odds) as additional features.

Using logistic regression is equivalent to weight learning in an MLN with no ontological constraints, since all facts are independent in the absence of the hard ontology formulas. Huynh and Mooney (2008) used a similar approach of learning weights for independent facts and adding in a hard transitivity constraint at inference time.

Alternatively, we can use standard gradient-based methods (Lowd and Domingos, 2007a) for Markov logic, but there are several issues we need to consider. First, they tend to be slow because the presence of hard constraints makes inference harder. Second, the query (i.e., non-evidence) atoms are all the candidate facts, and we only have labels for a small portion of them. So the

gradient of negative log-likelihood is replaced by

$$\frac{\partial \mathcal{L}(w|x,y)}{\partial w_i} = E_{w,y,h}[n_i(x,h,y)] - E_{w,h}n_i(x,h,y)$$

and the Hessian is replaced by

$$\begin{aligned}\frac{\partial^2}{\partial w_i \partial w_j}\mathcal{L}(w|x,y) = &(E_{w,y,h}[n_i n_j] - E_{w,y,h}[n_i]E_{w,y,h}[n_j]) \\ &- (E_{w,h}[n_i n_j] - E_{w,h}[n_i]E_{w,h}[n_j])\end{aligned}$$

With the incomplete training dataset, the objective function is no longer convex, and the gradient methods are not guaranteed to converge to the global minimum.

We use a method based on the diagonal Newton method presented in Lowd and Domingos (2007a). The weights are updated by

$$\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \alpha \boldsymbol{H}^{-1}\boldsymbol{g}$$

where α is the optimal step size. To speed up the method we use the persistent contrastive divergence (Murphy, 2012), and use MCSAT as inference algorithm with the sample size 10. Because we could have negative values on the diagonal of Hessian, we take the absolute values of them instead. The optimial step size α is also very difficult to estimate accurately, so we use a fixed α of 1.

3.3.5. Extensibility of Our Approach

A big advantage of our proposed model compared to other models is that it provides a general framework to combine information from different sources, as long as the information can be represented in first-order logic. Many ontologies are well

designed and properly reflects the necessary knowledge of specific domains, and all the knowledge or constraints are in the form of first-order logic. This suggests that our approach has very good extensibility.

For example, while the current ontology used in NELL is simple, in some ontologies, we may have more complex rules such as:

$$\textsc{Rel}(\texttt{city}, \texttt{country}, \texttt{CityCapitalOfCountry}) \wedge \texttt{city} \neq \texttt{city}' \Rightarrow \neg\textsc{Rel}(\texttt{city}', \texttt{country}, \texttt{CityCapitalOfCountry}).$$

which means there is only one capital for each country. Such formulas can easily be added into the model.

Some current extensions of NELL and similar IE systems can also be straightforwardly applied to our model. For instance, Lao et al. (2011) proposed an approach to learn the chain rules in NELL such as

$$\texttt{AthletePlaysForTeam}(x, y) \wedge \texttt{TeamPlaysInLeague}(y, z) \Rightarrow \texttt{AthletePlaysInLeague}(x, z)$$

These rules can be used to facilitate the system through inference by graph random walks. In Markov logic, this procedure can be viewed as a typical structural learning and MAP inference procedure. The formulas can be put into our model as:

$$\textsc{ChainRule}(r_1, r_2, r_3) \wedge \textsc{Rel}(x, y, r_1) \wedge \textsc{Rel}(y, z, r_2) \Rightarrow \textsc{Rel}(x, z, r_3)$$

with the evidence `ChainRule`(`AthletePlaysForTeam`, `TeamPlaysInLeague`, `AthletePlaysInLeague`).

We did not use any of these extensions in our experiments.

3.4. Experiment

3.4.1. Methodology

We evaluated our approach by applying it to the knowledge base extracted by NELL. We used the Markov logic formulas introduced in the previous sections. NELL's candidate instances, candidate extraction patterns, and seed instances were treated as evidence. Since NELL is a continuously running system, we took a snapshot for test. We used the 165th iteration as our dataset.

The instances that we chose for comparison spread over multiple predicates on several domains. Most predicates are from the sports domain, since this domain is widely used in NELL-related research for testing. For the comparison, we chose 13 relations and 10 categories. Each relation has about 1000-2000 instances and each category has about 5000-10000 instances. We randomly sampled about 200 instances for each category and relation, and 4511 in total as the test set. We also labeled 9887 instances from 5 relations and 6 categories as the training set.

Our system produces a list of all instances, ordered by marginal probability as computed by MC-SAT. We computed the precision, recall, and F1 score of our predictions by thresholding these probabilities, so that all facts with a probability of at least 0.5 were considered true, and all facts with a smaller probability were considered false. (We also explored using MaxWalkSAT for MAP inference, but found that it produced worse results.) For NELL, we evaluated precision, recall, and F1 score on its set of promoted facts.

Since NELL uses a semi-supervised bootstrap learning method, at each iteration it only promotes a limited number of high confidence instances into the KB in order to maintain high precision at the possible cost of lower recall.

Therefore we also compared the two methods using AUC. Our instances were ordered by their marginal probabilities. For NELL's result, we ordered promoted facts by the associated confidence values, followed by the rest of the candidate facts ordered by their associated confidences as well. This was necessary because NELL's confidence values for promoted and non-promoted facts are not comparable: some promoted facts have lower confidence than some non-promoted candidates. Naively ordering all facts by confidence value led to lower AUCs for NELL.

In order to see how the ontological constraints and pattern information help the joint inference, we experimented on several Markov logic networks.

- LR-NP: the logistic regression model with candidate and promoted facts as features (no extraction patterns or ontological constraints used);
- LR-NPO: LR-NP + MLN with ontological constraints for inference;
- LR-PS: LR-NP + extraction patterns confidence values trained with NELL's promoted facts;
- LR-PSO: LR-PS + MLN with ontological constraints for inference;
- LR-PSO*: LR-PSO with formula weights set to 1; no human labeled data is used in this model;
- LR-PL: same as LR-PS, but using human labels instead of NELL's promoted facts for training;
- LR-PLO: LR-PL + MLN with ontological constraints for inference;
- MLN-NPO: similar to LR-NPO, but trained with MLN instead of LR;
- MLN-PLO: similar to LR-PLO, but trained with MLN instead of LR.

We compare these methods with the original promoted facts of NELL and KGI-PSL (Pujara et al., 2013).

3.4.2. Results and Analysis

First, we show a brief comparison of the overall, all-category and all-relation performance of all the methods in Table 3.1 and Figure 3.1.

TABLE 3.1. Comparison of different methods on the NELL dataset

Method	AUC	AUC-Cat	AUC-Rel	Prec	Recall	F1
NELL	0.765	0.809	0.813	0.801	0.580	0.673
LR-NP	0.804	0.804	0.813	0.726	0.939	0.819
LR-PS	0.817	0.818	0.830	0.719	0.937	0.814
LR-PL	0.823	0.843	0.811	0.833	0.809	0.821
LR-NPO	0.874	0.918	0.828	0.736	**0.946**	0.828
LR-PSO	0.881	0.923	0.834	0.739	0.927	0.822
LR-PLO	**0.899**	**0.937**	**0.858**	**0.836**	0.837	**0.836**
LR-PSO*	0.840	0.912	0.827	0.694	0.751	0.721
MLN-NPO	0.892	N/A	N/A	0.784	0.893	0.835
MLN-PLO	0.866	N/A	N/A	0.799	0.864	0.830
PSL-KGI	0.904	N/A	N/A	0.777	0.944	0.853

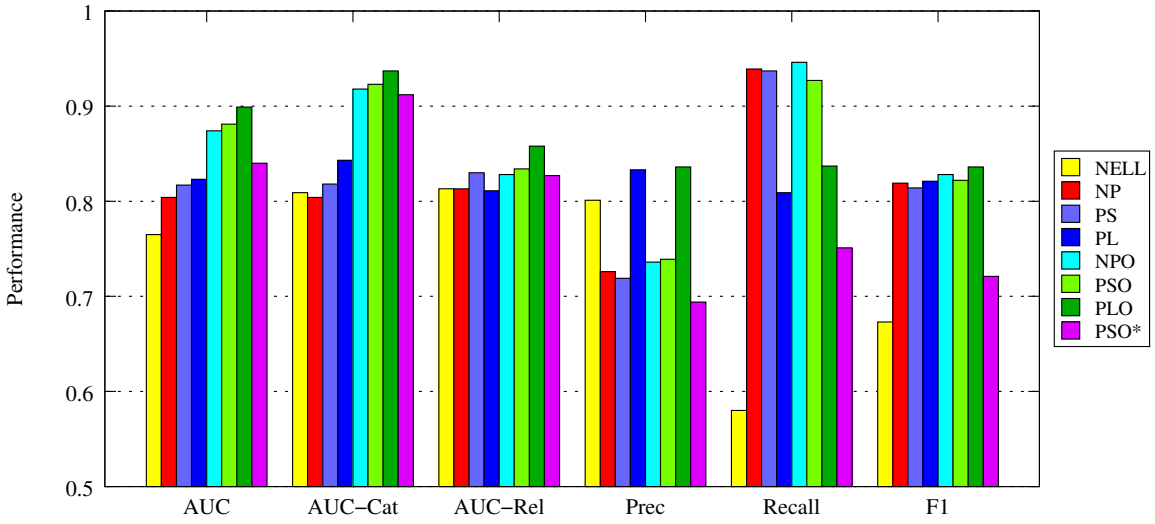

FIGURE 3.1. Comparison of different methods on the NELL dataset

Without the ontological constraints, there is no dependencies among instances, and our methods are equivalent to the logistic regression models. As we can see from Table 3.1, LR-NP, LR-PS and LR-PL achieve better AUCs and F1 than naively trusting NELL's promoted facts.

We then keep the weights learned with logistic regression, and add the ontological constraints at inference time to leverage the dependencies among the instances. All the three models with the ontological constraints (LR-NPO, LR-PSO, and LR-PLO) outperform their counterparts without ontological constraints.

The comparison of LR-NPO, LR-PSO and LR-PLO's results show that adding pattern information as an extra feature improves the overall performance. When the labeled training data are available, the results is even better than using NELL's promoted facts as the training data. However, the latter approach can be extended to any new categories or predicates without extra labels, while the former one needs labels in all the categories and predicates to train the pattern's logistic regression model.

We then evaluated the standard MLN learning methods MLN-NPO and MLN-PLO. MLN-NPO achieves comparable results to its LR counterpart LR-NPO, but MLN-PLO is worse than both LR-PLO and MLN-PLO. We find that the standard gradient-based learning methods are not very robust and effective on this task, and we will continue to investigate possible improvements on these methods for the future work.

Finally, we compared to PSL-KGI, a method very similar to ours. We used the results reported in Pujara et al. (2013). It has an AUC similar to LR-PLO, and has a better F1 but worse precision. In general, it also has a competitive performance.

It would also be interesting to look into the performances of individual predicates. Due to the limitation of space, we show the detailed overall and per-predicate performance only for NELL, LR-PSO* and LR-PLO in Table 3.2 and Figure 3.2. LR-PSO* does not use any labeled training data so it is perfectly fair to be compared with NELL, while LR-PLO is the best method with the training data.

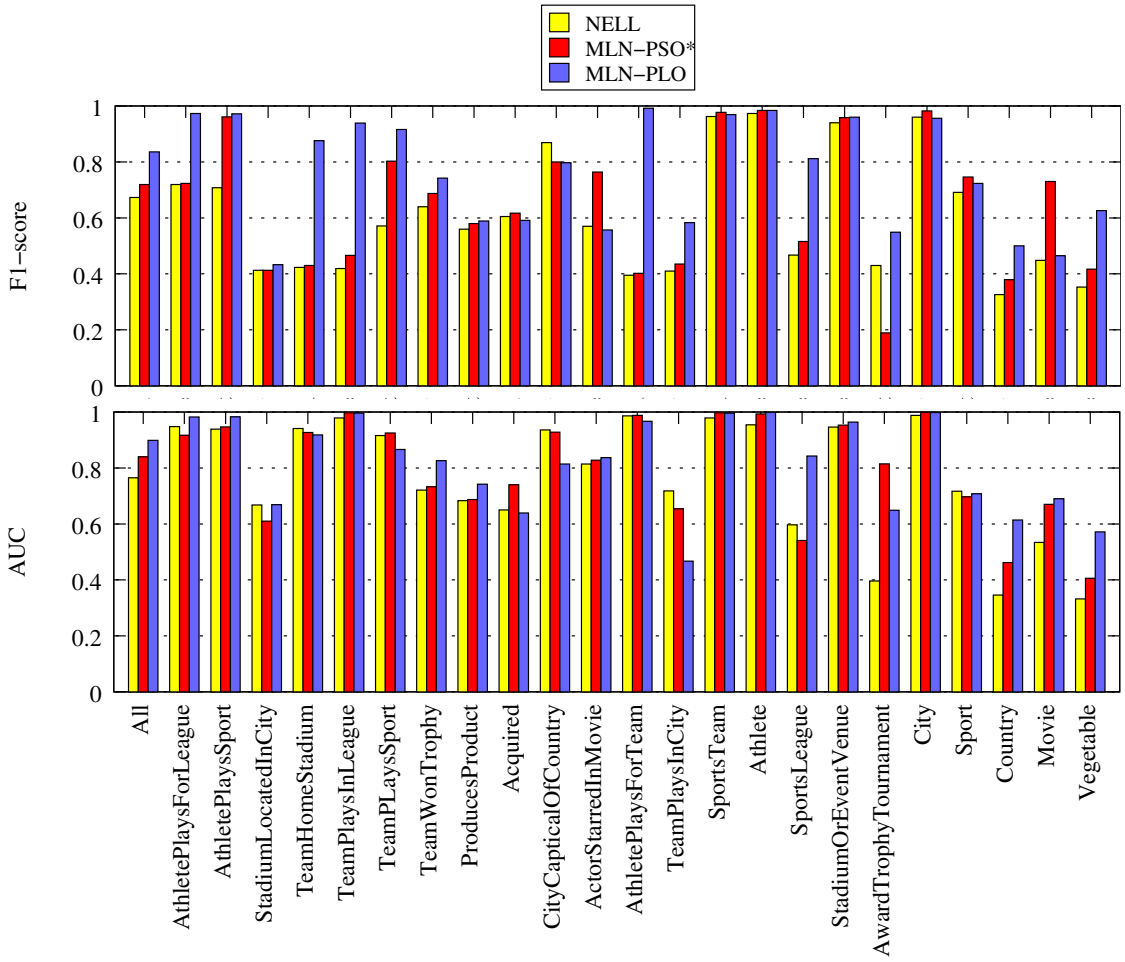

FIGURE 3.2. Comparison of NELL, LR-PSO* and LR-PLO on the NELL dataset by predicate based on F1 (above) and AUC (below)

As we can see from the table, LR-PLO has better F1 than NELL in 19 out of 23 predicates, and better AUCs in 16 out of 23 predicates. For the 8 relations and 4 categories that have no labeled training data, LR-PLO outperforms in 5 relations

TABLE 3.2. Comparison on the NELL dataset by predicate

	Precision			Recall			F1			AUC		
Predicate	NELL	PSO*	PLO	NELL	PSO*	PLO	NELL	PSO*	PLO	NELL	PSO*	PLO
All	0.801	0.745	**0.836**	0.580	0.696	**0.837**	0.673	0.719	**0.836**	0.765	0.840	**0.899**
Categories with training data												
Athlete	0.967	0.968	**0.978**	0.978	**1.000**	0.989	0.973	**0.984**	**0.984**	0.954	0.993	**0.999**
AwardTrophyTournament	0.288	0.105	**0.452**	0.850	**0.950**	0.700	0.430	0.189	**0.549**	0.396	**0.815**	0.649
City	0.975	**0.994**	**0.994**	0.946	**0.970**	0.922	0.960	**0.982**	0.956	0.988	**0.999**	0.998
SportsLeague	0.583	0.615	**0.929**	0.389	0.444	**0.722**	0.467	0.516	**0.812**	0.597	0.541	**0.843**
SportsTeam	0.953	0.955	**0.971**	0.972	**1.000**	0.967	0.962	**0.977**	0.969	0.979	**0.997**	0.996
StadiumOrEventVenue	0.925	0.919	**0.932**	0.956	**1.000**	0.989	0.940	0.958	**0.960**	0.946	0.953	**0.964**
Relations with training data												
AthletePlaysInLeague	0.930	0.930	**0.953**	0.586	0.591	**0.995**	0.719	0.723	**0.973**	0.948	0.917	**0.982**
AthletePlaysSport	**0.961**	0.940	0.946	0.560	0.983	**1.000**	0.708	0.961	**0.972**	0.939	0.947	**0.983**
StadiumLocatedInCity	**0.780**	**0.780**	0.649	0.281	0.281	**0.325**	0.413	0.413	**0.433**	0.668	0.610	**0.669**
TeamHomeStadium	**1.000**	**1.000**	0.849	0.268	0.274	**0.905**	0.423	0.430	**0.876**	**0.941**	0.927	0.918
TeamPlaysInLeague	0.960	0.965	**0.976**	0.268	0.307	**0.905**	0.419	0.466	**0.939**	0.979	**0.997**	0.996
Categories without training data												
Country	0.247	0.353	**0.800**	**0.477**	0.409	0.364	0.326	0.379	**0.500**	0.346	0.462	**0.614**
Movie	0.561	0.616	**0.698**	0.372	**0.895**	0.349	0.448	**0.730**	0.465	0.534	0.670	**0.690**
Sport	**0.644**	0.637	0.641	0.744	**0.899**	0.829	0.691	**0.746**	0.723	**0.717**	0.697	0.708
Vegetable	0.262	**0.271**	0.663	0.540	**0.900**	0.620	0.353	0.417	**0.626**	0.332	0.406	**0.572**
Relations without training data												
Acquired	0.750	0.755	**0.773**	0.507	**0.521**	0.479	0.605	**0.617**	0.591	0.650	**0.740**	0.639
ActorStarredInMovie	0.836	**0.839**	0.831	0.433	**0.702**	0.418	0.570	**0.764**	0.557	0.814	0.828	**0.837**
AthletePlaysForTeam	**1.000**	**1.000**	0.985	0.246	0.251	**1.000**	0.395	0.402	**0.992**	0.986	**0.988**	0.967
CityCapitalOfCountry	**0.955**	0.747	0.855	0.797	**0.861**	0.747	**0.869**	0.800	0.797	**0.936**	0.928	0.814
ProducesProduct	0.611	0.583	**0.615**	0.518	**0.576**	0.565	0.560	0.580	**0.589**	0.683	0.687	**0.742**
TeamPlaysInCity	0.862	**0.871**	0.438	0.269	0.290	**0.871**	0.410	0.435	**0.583**	**0.718**	0.654	0.467
TeamPlaysSport	**0.986**	0.846	0.866	0.402	0.765	**0.972**	0.571	0.803	**0.916**	0.916	**0.925**	0.866
TeamWonTrophy	**0.689**	0.581	0.612	0.597	0.840	**0.941**	0.640	0.687	**0.742**	0.721	0.733	**0.826**

and 4 categories for F1, and in 3 relations and 3 categories for AUC. LR-PSO* does somewhat better on the relations and categories with no labeled data, obtaining a higher AUC than NELL for 3 out of 4 categories and 6 out of 8 relations, and a higher F1 for all 4 categories and 7 out of 8 relations. Therefore, while both methods are effective, LR-PSO* appears to better generalize to new categories and relations since it does not rely on any training data.

Although the increases in precision and recall are modest, we are able to obtain them using only the information that NELL is already using. These gains are realized by replacing NELL's heuristic logical inference with a sound statistical relational approach that considers the joint uncertainty of many facts. The results show that our use of joint probabilistic inference is effective here.

3.4.3. Discussion

We may further look at some examples to see how exactly our approach refines the knowledge base and cleans the potential errors.

In the first example, `ProducesProduct` is a relation (predicate) whose domain is `Company` and range is `Product`. (`Adobe`, `Acrobat reader software`) and (`Adobe`, `Acrobat reader version`) are both candidate instances of `ProducesProduct` and have the same initial confidence. Our approach noticed that `Acrobat reader software` has a higher confidence value (thus higher probability) than `Acrobat reader version` to be an instance of `product`. Therefore it assigned a higher probability to the former relation instance than the latter one. NELL also uses type checking constraints, but its logical approach only allows the true relation instance to identify the true category instance, not vice versa. However, our Markov logic-based probabilistic framework can infer in both directions to achieve a better result.

Another example is that the entity `Los Angeles county` is extracted as an instance for both `City` and `County`. Although the former is wrong, it was extracted before the latter and got promoted by NELL since it had strong supporting evidence at that time. The latter also has supporting evidence, but it was not promoted because it violated the mutual exclusion rule of the two categories (i.e., a `City` cannot be a `County`, and vice versa). In this case, NELL's bootstrapping method tries to use the ontology constraints to rule out the wrong instances, but it fails when the wrong instances are promoted first. On the other hand, our joint inference framework is able to smartly reason about contradictory instances using all available information, rather than stubbornly enforcing earlier decisions.

3.5. Summary

We have proposed a method for cleaning an automatically extracted knowledge base using Markov logic. Our method uses probabilistic inference to simultaneously reason about the truth values of many related facts. This is an improvement on systems such as NELL, which uses logical inference and heuristics to update its knowledge base. Our proposed model is also a generic approach that can be extended with other sources of knowledge and constraints in first-order logic. Preliminary experiments show that our method achieves better F1 score and AUC than NELL's knowledge base. We also developed a custom local grounding method to make inference in this problem tractable. By learning weights for different matched patterns, we are able to create a confidence measure that is better calibrated than NELL's.

CHAPTER IV

KNOWLEDGE AWARE ONTOLOGY MATCHING

This work was published in the Proceedings of the 26th International Conference on Database and Expert Systems Applications (DEXA 2015). I was the primary contributor to the methodology and writing, and designed and conducted the experiments. The co-authors contributed partly to the methodology and writing. Dejing Dou and Daniel Lowd were the principle investigators for this work.

4.1. Introduction

Ontology matching is the process of aligning two semantically related ontologies. Traditionally, this task is performed by human experts from the domain of the ontologies. Since the task is tedious and error prone, especially in large ontologies, there has been substantial work on developing automated or semi-automated ontology matching systems (Shvaiko and Euzenat, 2011). While some automated matching systems make use of data instances (e.g., Doan et al. (2004)), in this dissertation we focus on the *schema-level* ontology matching task, in which no data instance is used.

Previous automatic ontology matching systems mainly use two classes of strategies. *Terminology-based* strategies discover corresponding concepts with similar names or descriptions. *Structure-based* strategies discover corresponding groups of concepts with similar hierarchies. In many cases, additional information about the relationships among the concepts is available through domain models, such as Bayesian networks, decision trees, and association rules. A domain model can be represented as a collection of *knowledge rules*, each of which denotes a

semantic relationship among several concepts. These relationships may be complex, uncertain, and rely on imprecise numeric values. In this dissertation, we introduce a new *knowledge-based strategy* which uses the structure of these knowledge rules as (soft) constraints on the alignment.

As a motivating example, consider two ontologies in the basketball game domain. One ontology has datatype properties `height`, `weight`, `center`, `forward` and `guard` for players, while the other ontology has the corresponding datatype properties `h`, `w`, and `position`. Terminology-based strategies may not identify these correspondences. However, if we know that a large value of `height` implies `center` is true in the first ontology, and the same relationship holds for `h` and `position` = `Center` in the second ontology, then we tend to believe that `height` maps to `h` and `center` maps to `position` = `Center`.

We use Markov logic networks (MLNs) as a probabilistic language to combine the knowledge-based strategy with other strategies, in a formalism similar to that of Niepert et al. (2010). In particular, we encode the knowledge-based strategy with weighted formulas that increase the probability of alignments where corresponding concepts have isomorphic relationships. We use an MLN inference engine to find the most likely alignment. We name our method Knowledge-Aware Ontology Matching (KAOM).

Our approach is also capable of identifying *complex correspondences*, an extremely difficult task in ontology matching. A complex correspondence is a correspondence between a simple concept and a complex concept (e.g., `grad_student` maps to the union of `PhD` and `Masters`). This can be achieved by constructing a set of *complex concepts* (e.g., unions of concepts) in each ontology,

subsequantly generating candidate complex correspondences, and using multiple strategies – including the knowledge-based strategy – to find the correct ones.

The chapter is organized as follows. In Section 4.2., we define ontology matching and review previous work. In Section 4.3., we introduce the concept of "knowledge rules" with a definition and examples. In Section 4.4., we present the knowledge-based strategy. In Section 4.5., we show how to incorporate complex concepts in our method. In Section 4.6., we formalize our method with Markov logic networks. We present experimental results in Section 4.7. and conclude in Section 4.8..

4.2. Ontology Matching

Most existing schema-level ontology matching systems use two types of strategies: terminology-based and structure-based. Terminology-based strategies are based on terminological similarity, such as string-based or linguistic similarity measures. Structure-based strategies are based on the assumption that two matching ontologies should have similar local or global structures, where the structure is represented by subsumption relationships of classes and properties, and domains and ranges of properties. Advanced ontology matching systems often combine the two types of strategies (Cotterell and Medina, 2013; Mao et al., 2010; Melnik et al., 2002; Noy and Musen, 2000). See Shvaiko and Euzenat (2011) for a survey of ontology matching systems and algorithms.

Recently, a probabilistic framework based on Markov logic was proposed to combine multiple strategies (Niepert et al., 2010). In particular, it encodes multiple strategies and heuristics into hard and soft constraints, and finds the best matching by minimizing the weighted number of violated constraints. The

constraints include string similarity, the cardinality constraints which enforce that each concept matches at most one concept, the coherence constraints which prevent inconsistency induced by the matching, and the stability constraints which penalize dissimilar local subsumption relationships.

Previous work has taken several different approaches to find complex correspondences (i.e., complex matching). Dhamankar et al. (2004) construct candidates for complex correspondences using operators for primitive classes, such as string concatenation or arithmetic operations on numbers. Ritze et al. (2008) summarize four patterns for building up complex correspondences based on linguistic and structural features given a candidate one-to-one alignment: Class by Attribute Type (CAT), Class by Inverse Attribute Type (CIAT), Class by Attribute Value (CAV), and Property Chain pattern (PC). Finally, when aligned or overlapping data is available, inductive logic programming (ILP) techniques can be used as well (Hu et al., 2011; Qin et al., 2007).

Many ontology matching systems make use of data instances to some extent (e.g., (Dhamankar et al., 2004; Doan et al., 2002; Hu et al., 2011; Qin et al., 2007)). However, in this dissertation, we focus on the case where data are not available or data sharing is not preferred because of communication cost or privacy concerns.

4.3. Representation of Domain Knowledge

In an OWL ontology, knowledge is represented as a set of DL axioms. These axioms describe properties of classes or relations (e.g., a relation is functional, symmetric, or antisymmetric, etc.), or a relationship of several classes or relations (e.g., the relation 'grandfather' is the composition of the two relations 'father' and 'parent').

The choice of DL as the foundation of the Semantic Web ontology languages is largely due to the trade-off between expressivity and reasoning efficiency. In tasks such as ontology matching, reasoning does not need to be instant, so we can afford to consider other forms of knowledge outside of a specific ontology language or description logic.

Definition 4.1 (Knowledge Rule)**.** A knowledge rule is a sentence $R(a, b, \ldots; \theta)$ in a formal language which consists of a relation R, a set of entities (i.e., classes, attributes or relations) $\{a, b, \ldots\}$, and (optionally) a set of parameters θ. A knowledge rule carries logical or probabilistic semantics representing the relationship among these entities. The specific semantics depend on R.

Many domain models and other types of knowledge can be represented as sets of knowledge rules, each rule describing the relationship of a small number of entities. The semantics of each relationship R can typically be expressed with a formal language. Table 4.1 shows some examples of the symbols used in formal languages such as description logic, along with their associated semantics.

TABLE 4.1. Syntax and semantics of DL axioms (top), and other non-DL knowledge rules used in the examples (bottom)

Description	Syntax	Semantics
Subsumption	$C \sqsubseteq D$	$C^{\mathcal{I}} \subseteq D^{\mathcal{I}}$
Equivalence	$C \equiv D$	$C^{\mathcal{I}} = D^{\mathcal{I}}$
Disjointness	$C \sqsubseteq \neg D$	$C^{\mathcal{I}} \cap D^{\mathcal{I}} = \emptyset$
Precedence	$R \prec S$	$y < y'$ for $\forall (x, y) \in R^{\mathcal{I}} \wedge (x, y') \in S^{\mathcal{I}}$
Probabilistic rule	$C \Rightarrow D$	$\Pr(D^{\mathcal{I}} \mid C^{\mathcal{I}})$ is close to 1

We illustrate a few forms of knowledge rules with the following examples. For each rule, we provide a description in English, a logical representation, and an encoding as a knowledge rule with a particular semantic relationship, R_i. We

define a new relationship in each example, but, in a large domain model, most relationships would be appear many times in different rules.

Example 4.1. The submission deadline precedes the camera ready deadline:

$$\texttt{paperDueOn} \prec \texttt{manuscriptDueOn}$$

This is represented as $R_1(\texttt{paperDueOn}, \texttt{manuscriptDueOn})$ with $R_1(a, b) : a \prec b$.

Example 4.2. A basketball player taller than 81 inches and heavier than 245 pounds is likely to be a center:

$$\texttt{h} > 81 \wedge \texttt{w} > 245 \Rightarrow \texttt{pos} = \texttt{Center}$$

This rule can be viewed as a branch of a *decision tree* or an *association rule*. It can be represented as $R_2(\texttt{h}, \texttt{w}, \texttt{pos=Center}, [81, 245])$, with $R_2(a, b, c, \theta) : a > \theta_1 \wedge b > \theta_2 \Rightarrow c$.

Example 4.3. A smoker's friend is likely to be a smoker as well:

$$\texttt{Smokes}(x) \wedge \texttt{Friend}(x, y) \Rightarrow \texttt{Smokes}(y)$$

Relational (i.e., first-order logic) rules such as this one describe relationships of attributes across multiple tables, as opposed to propositional data mining rules that are restricted to a single table. This rule can be represented as $R_3(\texttt{Smoke}, \texttt{Friend})$ with $R_3(a, b) : a(x) \wedge b(x, y) \Rightarrow a(y)$. In fact, all DL axioms are merely syntax sugars for first-order logic rules. For example, in a previous example, the DL axiom $\texttt{grandfather} \equiv \texttt{father} \circ \texttt{parent}$ represented in our universal knowledge rule

representation $R_{3'}(\texttt{grandfather}, \texttt{father}, \texttt{parent})$ with $R_{3'}(a, b, c) : a \equiv b \circ c$ is basically a first-order rule

$$\exists y \texttt{father}(x, y) \wedge \texttt{parent}(y, z) \Leftrightarrow \texttt{grandfather}(x, z).$$

For the remainder of this chapter, we will assume that the knowledge in both domains is represented as knowledge rules, as described in this section.

4.4. Our New Knowledge-Based Strategy

We propose a new strategy for ontology matching that uses the similarity of knowledge rules in the two ontologies. It is inspired by the structure-based strategy in many ontology matching algorithms (e.g., (Melnik et al., 2002) and (Niepert et al., 2010)). It naturally extends the subsumption relationship of entities in structure-based strategies to other types of relationships.

We use Markov logic to combine the knowledge-based strategy with other strategies. In particular, each strategy is represented as a set of *soft constraints*, each of which assigns a score to the alignments satisfying it, and the alignment with the highest total score is chosen as the best alignment. We now describe the soft constraints encoding the knowledge-based strategy. Our complete Markov logic-based approach, including the soft constraints required for the other strategies, will be described in Section 4.6..

For each relation R_k that appears in both domains, we introduce a set of soft constraints so that the alignments that preserve these relationships are preferred to

those that do not:

$$+w_k \qquad R_k(a,b) \wedge \neg R_k(a',b') \Rightarrow a \not\equiv a' \vee b \not\equiv b'$$

$$+w'_k \qquad R_k(a,b) \wedge R_k(a',b') \Rightarrow a \equiv a' \wedge b \equiv b'$$

$$\forall a, b \in O_1, a', b' \in O_2$$

These formulas assume R_k is a binary relation, but they trivially generalize to any arity, e.g., $R_k(a, b, c, d, e, \ldots)$. Note that separate constraints are created for each possible tuple of constants from the respective domains. The numbers preceding the constraints (w_k and w'_k) are the *weights*. A larger weight represents a stronger constraint, since alignments are ranked based on the total weights of the constraints they satisfy. A missing weight means the constraint is a hard constraint which must be satisfied.

Example 4.4. A reviewer of a paper cannot be the paper's author. In the `cmt` [1] ontology we have $R_4(\texttt{writePaper}, \texttt{readPaper})$ and in the `confOf` ontology we have $R_4(\texttt{write}, \texttt{reviews})$ where $R_4(a, b) : a \sqsubseteq \neg b$ is the disjoint relationship of properties. Applying the constraint formulas defined above, we increase the score of all alignments containing the two correct correspondences: `writePaper` $\equiv$ `writes` and `readPaper` $\equiv$ `reviews`.

Rules involving continuous numerical attributes often include parameters (e.g., thresholds in Example 4.2) that do not match between different ontologies. In order to apply the knowledge-based strategy to numerical attributes, we make the assumption that corresponding numerical attributes roughly have a *positive linear*

[1]Throughout this chapter, we will use ontologies in the conference domain (`cmt`, `confOf`, `conference`, `edas`, `ekaw`) and the NBA domain (`nba-os`, `yahoo`) in our examples. The characteristics of these ontologies will be further described in Section 4.7..

transformation. This assumption is often true in real applications, for instance, when an imperial measure of height matches to a metric measure of height. We propose two methods to handle numerical attributes.

The first method is to compute a *distance measure* (e.g., Kullback-Leibler divergence) between the distributions of the corresponding attributes in a candidate alignment. Although the two distributions describe different attributes, the distance can be computed by assuming a linear transformation between the two attributes. The coefficients of the mapping relation can be roughly estimated using the ranges of attribute values appearing in the knowledge rules (see Example 4.5 below).

Specifically, if the distance between rules $R(\mathtt{a}, \mathtt{b}, \ldots, \theta)$ and $R(\mathtt{a'}, \mathtt{b'}, \ldots, \theta')$ is d, then we add the constraint:

$$a \equiv a' \wedge b \equiv b' \wedge c \equiv c'$$

with a weight of $\max(d_0 - d, 0)$ for a given threshold d_0.

Example 4.5. In the `nba-os` ontology, we have conditional rules converted from a decision tree, such as

$$\mathtt{h} > 81 \wedge \mathtt{w} > 245 \Rightarrow \mathtt{Center}$$

Similarly, in the `nbayahoo` ontology, we have

$$\mathtt{h'} > 2.06 \wedge \mathtt{w'} > 112.5 \Rightarrow \mathtt{Center'}$$

Here the knowledge rules represent the conditional distributions of multiple entities. We define the distance between the two conditional distributions as

$$d(\mathtt{h}, \mathtt{w}, \mathtt{Center}; \mathtt{h'}, \mathtt{w'}, \mathtt{Center'}) = \mathbb{E}_{p(\mathtt{h},\mathtt{w})} d(p(\mathtt{Center}|\mathtt{h}, \mathtt{w})||p(\mathtt{Center'}|\mathtt{h'}, \mathtt{w'}))$$

where $\mathbb{E}(\cdot)$ is expectation and $d(p||p')$ is a distance measure. Because `Center` and $\mathtt{Center}'$ are binary attributes, we simply use $|p - p'|$ as the distance measure. For numerical attributes, we can use the difference of two distribution histograms as the distance measure. We assume the attribute correspondences (`h` and `h'`, `w` and `w'`) are linear mappings, and the linear relation can be roughly estimated (e.g., by simply matching the minimum and maximum numbers in these rules). When computing the expectation over `h` and `w`, we apply the linear mapping to generate corresponding values of `h'` and `w'`, e.g., `h'` $= 0.025$ `h`, `w'` $= 0.45$ `w`. The distribution of the conditional attributes $p(\mathtt{h}, \mathtt{w})$ can be roughly estimated as independent and uniform over the ranges of the attributes.

The second method for handling continuous attributes is to discretize them, reducing the continuous attribute problem to the discrete problem described earlier. For example, suppose each continuous attribute x is replaced with a discrete attribute x^d, indicating the quartile of x rather than its original value. Then we have $R_5(\mathtt{h}^d, \mathtt{w}^d, \mathtt{Center})$ and $R_5(\mathtt{h'}^d, \mathtt{w'}^d, \mathtt{Center'})$ with relation $R_5(a, b, c) : a = 4 \wedge b = 4 \Rightarrow c$, and the discrete value of 4 indicates that both a and b are in the top quartile. Other discretization methods are also possible, as long as the discretization is done the same way in both domains.

Our method does not rely on the forms of knowledge rules, nor does it rely on the algorithms used to learn these rules. As long as similar techniques or tools

are used on both sides of ontologies, we would always be able to find interesting knowledge-based similarities between the two ontologies.

4.5. Finding Complex Correspondences

Our approach can also find complex correspondences, which contain complex concepts in either or both of the ontologies. We add the complex concepts into consideration and treat them the same way as simple concepts, and then we jointly solve all the simple and complex correspondences by considering terminology, structure, and knowledge-based strategies in a single probabilistic formulation.

First, because complex concepts are recursively defined and potentially infinite, we need to select a finite subset of complex concepts and use them to generate the candidate correspondences. We will only include the complex concepts occurring in the ontology axioms or in the knowledge rules.

Second, we need to define a string similarity measure for each type of complex correspondence. For example, Ritze et al. (2008) requires two conditions for a Class by Attribute Type (CAT) matching pattern $O_1 : a \equiv O_2 : \exists p.b$ (e.g., $a =$ `Accepted_Paper`, $p =$ `hasDecision`, $b =$`Acceptance`): a and b are terminologically similar, and the domain of p (`Paper` in the example) is a superclass of a. We can therefore define the string similarity of a and $\exists p.b$ to be the string similarity of a and b which coincides with the first condition, and the second condition is encoded in the structure stability constraints. The string similarity measure of many other types of correspondences can be defined similarly based on the heuristic method in Ritze et al. (2008). If there does not exist a straight-forward way to define the string similarity for a certain type of complex correspondences, we can simply set it to 0 and rely on other strategies to identify such correspondences.

Lastly, we need constraints for the correspondence of two complex concepts. The corresponding component concepts and same constructor always implies the corresponding complex concepts, while in the other direction, it is a soft constraint.

$$\text{cons}_k(a,b) \equiv \text{cons}_k(a',b') \Leftarrow a \equiv a' \wedge b \equiv b'$$

$$+w_k^c \qquad \text{cons}_k(a,b) \equiv \text{cons}_k(a',b') \Rightarrow a \equiv a' \wedge b \equiv b'$$

where cons_k are different constructors for complex concepts, e.g., union, $\exists p.b$.

Some complex correspondences are almost impossible to be identified with traditional strategies. With the knowledge-based strategy, it becomes possible.

Example 4.6. A reviewer of a paper cannot be the paper's author. In the `cmt` ontology we have

$$\texttt{writePaper} \sqsubseteq \neg\texttt{readPaper}$$

and in the `conference` ontology we have

$$\texttt{contributes} \upharpoonright \texttt{Reviewed\_contribution} \sqsubseteq \neg(\texttt{contributes} \circ \texttt{reviews})$$

We first build two complex concepts $\texttt{contributes} \upharpoonright \texttt{Reviewed\_contribution}$ and $\texttt{contributes} \circ \texttt{reviews}$. With $R_4(a,b) = a \sqsubseteq \neg b$ (disjoint properties), the score function would favor the correspondences

$$\texttt{writePaper} \equiv \texttt{contributes} \upharpoonright \texttt{Reviewed\_contribution}$$

$$\texttt{readPaper} \equiv \texttt{contributes} \circ \texttt{reviews}$$

4.6. Knowledge Aware Ontology Matching

In this section, we present our approach, Knowledge Aware Ontology Matching (KAOM). KAOM uses Markov logic networks to solve the ontology matching task. The MLN formulation is similar to Niepert et al. (2010) but incorporates the knowledge-based matching strategy and treatment of complex correspondences.

We represent a correspondence using a binary relation, `match(a1,a2)`, which is true if concept `a1` from the first ontology is semantically equivalent to concept `a2` from the second ontology (e.g., `match(writePaper, writes)` means `writePaper` $\equiv$ `writes`). Each possible world therefore corresponds to an alignment of the two ontologies. We want to find the most probable possible world, which is the configuration that maximizes the sum of weights of satisfied formulas.

We define three components of the MLN of the ontology matching problem: *constants*, *evidence* and *formulas*. The constants are the entities in both ontologies, including the simple named ones and the complex ones. The evidence includes the complete set of OWL-supported relationships (e.g., subsumptions and disjointness) among all concepts in each ontology, and rules represented as first-order atomic predicates as described in the Section 4.3.. We use an OWL reasoner to create the complete set of OWL axioms.

For the formulas, we begin with a set of formulas adapted from Niepert et al. (2010):

1. *A-priori similarity* is the string similarity between all pairs of concepts:

$$s_{a,a'} \quad \texttt{match}(a, a')$$

where $s_{a,a'}$ is the string similarity between a and a', which also serves as the weight of the formula. We use the Levenshtein measure (Levenshtein, 1966) for simple correspondences. This atomic formula increases the probability of matching pairs of concepts with similar strings, all other things being equal.

2. *Cardinality constraints* enforce one-to-one simple (or complex) correspondences:

$$\texttt{match}(a, a') \wedge \texttt{match}(a, a'') \Rightarrow a' = a''$$

3. *Coherence constraints* enforce consistency of subclass relationships:

$$\texttt{match}(a, a') \wedge \texttt{match}(b, b') \wedge a \sqsubseteq b \Rightarrow a' \sqsubseteq \neg b'$$

4. *Stability constraints* enforce consistency of the subclass relationships between the two ontologies. They can be viewed as a special case of the knowledge-based constraints we introduce below.

4.6.1. Knowledge-based Constraints

We now describe how we incorporate knowledge-based constraints into the MLN formulation through new formulas relating knowledge rules to matchings. The *stability* constraints in Niepert et al. (2010) consider three subclass relationships, including a is a subclass of b (`subclass`), and a is a subclass or superclass of the domain or range of a property b (`domainsub`, `rangesub`). We extend the relationships (knowledge rule patterns) to sub-property, disjoint properties, and user-defined relations such as ordering of dates, and non-deterministic relationships

such as correlation and anti-correlation:

$$-w_k \qquad R_k(a,b,...) \wedge \neg R_k(a',b',...) \Rightarrow \texttt{match}(a,a') \wedge \texttt{match}(b,b') \wedge ..., k = 1,...,m \tag{Equation 4.1}$$

where m is the number of knowledge rule patterns. User-defined relations include those derived from decision trees, association rules, expert systems, and other knowledge sources outside the ontology.

Besides the stability constraints, we introduce a new group of *similarity* constraints that encourage knowledge rules with the same pattern to have corresponding concepts.

$$+w'_k \qquad R_k(a,b,...) \wedge R_k(a',b',...) \Rightarrow \texttt{match}(a,a') \wedge \texttt{match}(b,b') \wedge ..., k = 1,...,m \tag{Equation 4.2}$$

For numerical rules, we instead use MLN formulas:

$$d_0 - d \quad \texttt{match}(a,a') \wedge \texttt{match}(b,b') \wedge ..., k = 1,...,m \tag{Equation 4.3}$$

where d is a distance measure of the two rules $R_k(a,b,...)$ and $R'_k(a',b',...)$ and d_0 is a threshold determining whether the rules are similar or not.

To handle complex correspondences, we add complex concepts that occur in knowledge rules as constants of the MLN, and add knowledge rules that contain these new complex concepts. We define the string similarity and enforce type constraints between simple and complex concepts, as described in Section 4.5.. For complex to complex correspondences, the string similarity measure is zero, but we

have constraints

$$\texttt{match}(a, a') \wedge \texttt{match}(b, b') \wedge ... \Rightarrow \texttt{match}(c, c')$$

$$w_k^c \qquad \texttt{match}(a, a') \wedge \texttt{match}(b, b') \wedge ... \Leftarrow \texttt{match}(c, c')$$

where $c = \text{cons}_k(a, b, ...), c' = \text{cons}_k(a', b', ...)$ for each constructor cons_k.

4.7. Experiments

We test our KAOM approach on three domains: NBA, census, and conference. The sizes of the ontologies of these domains are listed in Table 4.2. These domains contain very different forms of ontologies and knowledge rules, so we can examine the generality and robustness of our approach.

TABLE 4.2. Profile of the datasets, including the number of classes, object properties, data properties and nominal values of each ontology.

domain	ontology	# classes	# object props	# data props	# values
NBA	nba-os	3	3	20	3
	yahoo	4	4	21	7
census	adult	1	0	15	101
	income	1	0	12	97
OntoFarm	cmt	36	50	10	0
	confOf	38	13	25	0
	conference	60	46	18	6
	edas	103	30	20	0
	ekaw	78	33	0	0

We use Pellet (Sirin et al., 2007) for logical inference of the ontological axioms and TheBeast[2] (Riedel, 2008) and Rockit[3] (Noessner et al., 2013) for Markov logic inference. We ran all experiments on a machine with 24 Intel Xeon E5-2640 cores

[2] `http://code.google.com/p/thebeast/`

[3] `https://code.google.com/p/rockit/`. We use RockIt for the census domain because TheBeast is not able to handle the large number of rules in that domain.

@2500 MHz and 8GB memory. We compare our system (KAOM) with three others: KAOM without the knowledge-based strategy (MLOM), CODI (Huber et al., 2011) (a different implementation of MLOM), and logmap2 (Jiménez-Ruiz et al., 2012), a top performing system in OAEI 2014 [4].

We manually specify the weights of the Markov logic formlas in KAOM and MLOM. The weights of stability constraints for subclass relationships are set with values same as the ones used in (Niepert et al., 2010), i.e., the weight for subclass is -0.5, and those for sub-domain and range are -0.25. In KAOM, we also set the weights for different types of similarity rules based on our assessment of their relative importance and kept these weights fixed during the experiments.

4.7.1. NBA

The NBA domain is a simple setting that we use to demonstrate the effectiveness of our approach. We collected data from the NBA official website and the Yahoo NBA website. For each ontology, we used the WinMine toolkit [5] to learn a decision tree for each attribute using the other attributes as inputs.

For each pair of conditional distributions based on decision tree with up to three attributes, we calculate their similarity based on the distance measure described in Example 4.5. We use the Markov logic formula (Equation 4.3) with the threshold $d_0 = 0.2$. To make the task more challenging, we did not use any name similarity measures. Our method successfully identified the correspondence of all the numerical and nominal attributes, including height, weight and positions

[4] `http://oaei.ontologymatching.org/2014/`

[5] `http://research.microsoft.com/en-us/um/people/dmax/WinMine/Tooldoc.htm`

(center, forward and guard) of players. In contrast, without a name similarity measure, no other method can solve the matching problem at all.

4.7.2. Census

We consider two census datasets and their ontologies from UC Irvine data repository[6]. Both datasets represent census data but are sampled and post-processed differently. These two census ontologies are flat with a single concept but many datatype properties and nominal values. For this domain, we use association rules as the knowledge. We first discretize each numerical attribute into five intervals, and then generate association rules for each ontology using the Apriori algorithm with a minimum confidence of 0.9 and minimum support of 0.001. For example, one generated rule is:

```
age='(-inf-25.5]' education='11th' hours-per-week='(-inf-35.5]'
  ==> adjusted-gross-income='<=50K' conf:(1)
```

This is represented as

$$R_6(\texttt{age}^d, \texttt{11th}, \texttt{hours-per-week}^d, \texttt{adjusted-gross-income}^d)$$

where x^d refers to the discretized value of x, split into one fifth percentile intervals, and $R_6(a, b, c, d) : a = 1 \wedge b \wedge c = 1 \Rightarrow d = 1$. For scalability reasons, we consider up to three concepts in a knowledge rule, i.e., association rules with up to three attributes. We set the weight of knowledge similarity constraints for the association rules to 0.25.

[6] `https://archive.ics.uci.edu/ml/datasets.html`

In the Markov logic formulation in Niepert et al. (2010), only the correspondences with apriori similarity measure larger than a threshold τ are added as evidence. In the experiments, we set τ with different values from 0.50 to 0.90. When τ is large, we deliberately discard the string similarity information for some correspondences. MLOM for this task is an extension of Niepert et al. (2010) by adding correspondences of *nominal values* and their dependencies with the related attributes. The results are shown in Figure 4.1. We can see that KAOM always gets better recall and F1, with only a slight degradation in precision. This means our approach fully leverages the knowledge rule information and thus does not rely too much on the names of the concepts to determine the matching. For example, when τ is 0.70, KAOM finds 6 out of 8 correspondences of values of `adult:workclass` and `income:class_of_worker`, while MLOM finds none. The other two systems were not designed for nominal value correspondences. CODI only finds 7 and logmap2 only finds 3 attribute correspondences, while KAOM and MLOM find all the 12 attribute correspondences.

4.7.3. OntoFarm

In order to show how our system can use manually created expert knowledge bases, we use OntoFarm, a standard ontology matching benchmark for an academic conference domain as the third domain in our experiments. As part of OAEI, it has been widely used in the evaluation of ontology matching systems. The process of manually knowledge rule creation is time consuming, so we only used 5 of the OntoFarm ontologies (`cmt`, `conference`, `confOf`, `edas`, `ekaw`). Using their knowledge of computer science conferences and the structure of just one ontology, two individuals listed a number of rules (e.g., Example 4.1). We then translated

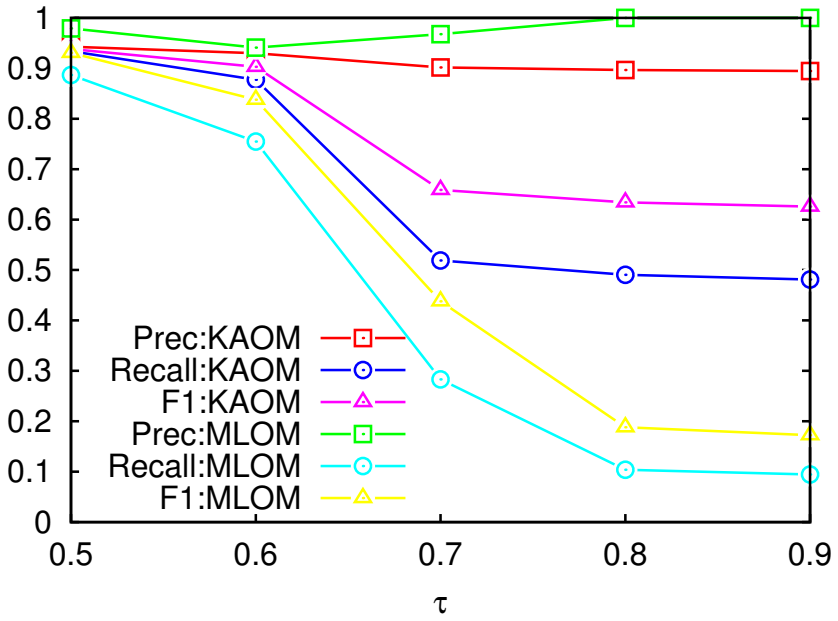

FIGURE 4.1. Precision, recall and F1 on the census domain as a function of the string similarity threshold τ

these rules into each of the five ontologies. Thus, the same knowledge was added to each of the ontologies, but its representation depended on the specific ontology. For some ontologies, some of the rules were not representable with the concepts in them and thus had to be omitted. This manually constructed knowledge base was developed before running any experiments and kept fixed throughout our experiments. Among the 5 ontologies, we have 10 pairs of matching tasks in total. We set τ to 0.70, and the weight for the knowledge similarity constraints to 1.0.

We first compare the four methods to the reference one-to-one alignment from the benchmark (Figure 4.2). KAOM achieves similar precision and F1, and better recall than other systems. It was able to identify correspondences in which the concept names are very different, for instance, `cmt:readPaper` $\equiv$ `confOf:reviews`. Note that the similarity constraints work in concert with other constraints. For instance, in Example 4.4, since disjointness is a symmetric knowledge rule, domain

and range constraints could be helpful to identify whether `cmt:writePaper` should match to `confOf:writes` or `confOf:reviews`.

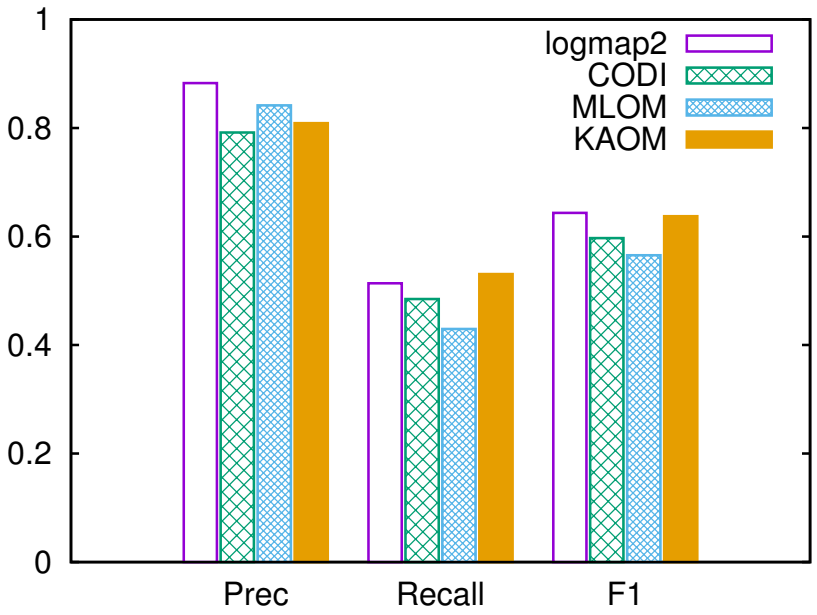

FIGURE 4.2. Precision, recall and F1 on the OntoFarm domain with only the one-to-one correspondences

To evaluate our approach on complex correspondences, we extended the reference alignment with hand-labeled complex correspondences (Figure 4.3). MLOM does not perform well in this task because the complex correspondences require a good similarity measure to become candidates (such as the linguistic features in Ritze et al. (2008)). KAOM, however, uses the structure of the rules to find many complex correspondences without relying on complex similarity measures. For this task we also tried learning the weights of the formulas [7] (KAOM-learn). For each of the 10 pairs of ontologies, we used the rest 9 pairs as training data. KAOM-learn performs slightly better than KAOM.

[7]We used MIRA implemented in TheBeast for weight learning.

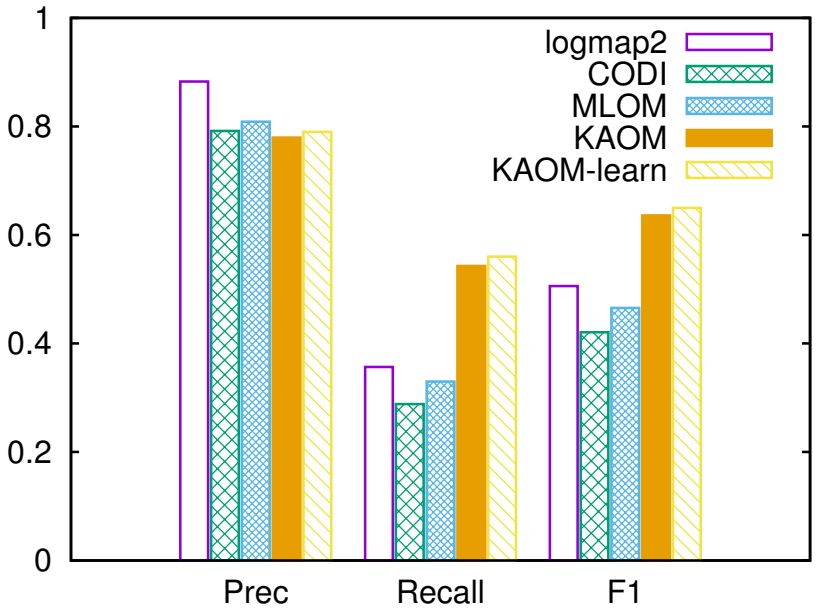

FIGURE 4.3. Precision, recall and F1 on the OntoFarm domain with the complex correspondences

With the hand-picked or automatically learned weights, KAOM produces a single most-likely alignment. However, we can further tune KAOM to produce alignments with higher recall or higher precision. We accomplish this by adding the MLN formula $\texttt{match}(a, a')$ with weight w. When w is positive, alignments with more matches are more likely, and when w is negative, alignments with fewer matches are more likely (all other things being equal). We adjusted this weight to produce the precision-recall curve shown in Figure 4.4. KAOM dominates CODI and provides much higher recall values than logmap2, although logmap2's best precision remains slightly above KAOM's.

4.8. Summary

We proposed a new ontology matching algorithm KAOM. The key component of KAOM is the knowledge-based strategy, which is based on the intuition that

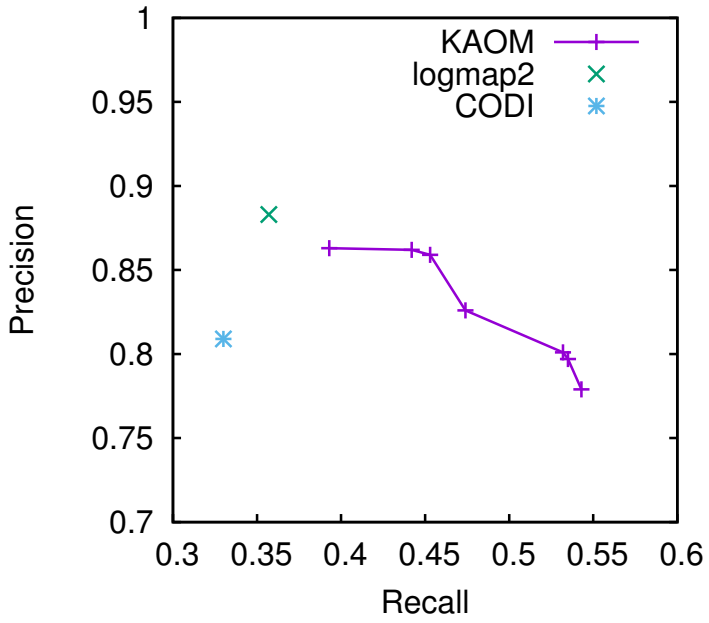

FIGURE 4.4. Precision-recall curve on the OntoFarm domain with the complex correspondences

ontologies about the same domain should contain similar knowledge rules, in spite of the different terminologies they use. KAOM is also capable of discovering complex correspondences, by treating complex concepts the same way as simple ones. We encode the knowledge-based strategy and other strategies in Markov logic and find the best alignment with its inference tools. Experiments on the datasets and ontologies from three different domains show that our method effectively uses knowledge rules of different forms to outperform several state-of-the-art ontology matching methods.

CHAPTER V

A PROBABILISTIC APPROACH TO KNOWLEDGE TRANSLATION

This work is to appear in the Proceedings of the 30th AAAI Conference on Artificial Intelligence (AAAI-16). I was the primary contributor to the methodology and writing, and designed and conducted the experiments. The co-authors contributed partly to the methodology and writing. Dejing Dou and Daniel Lowd were the principle investigators for this work.

5.1. Introduction

Knowledge acquisition is a critical process for building predictive or descriptive models for many applications. When domain expertise is available, knowledge can be constructed manually. When enough high-quality data is available, knowledge can be constructed automatically using data mining or machine learning tools. Both approaches can be difficult and expensive, so we would prefer to reuse or transfer knowledge from one application or system to another whenever possible. However, different applications or systems often have different representations, which makes knowledge reuse or transfer a non-trivial task. For example, genetic databases normally use different schemas to store genotype and phenotype data from distinct NIH model organisms. Different EEG (electroencephalography) labs use their own terminology in spreadsheets for recording experiments. Online bookstores may use different XML schemas to describe commodity, transaction, and shipping information.

As a motivating example, suppose a new credit card company without historical data wants to use the classification model mined by a partner credit card

company to determine whether the applicants of the new company are qualified or not. Since the two companies may use different schemas to store their applicants' data (e.g., in one schema, we have annual income recorded as a numerical attribute, while in the other, we have salary as an attribute with discretized ranges), we cannot simply reuse the old classifier. Due to privacy and scalability concerns, we cannot translate the collaborative company's data to the new schema either. Finally, since the new company is new, it may not yet have any data to learn a model under its new schema. Therefore, we want to *translate* the classification model itself to the new schema, *without using any data*.

We propose *knowledge translation* (KT) as a novel solution to translate knowledge across conceptually similar but semantically heterogeneous schemas or ontologies. For convenience, we refer to them generically as "schemas." We define the knowledge translation task, propose specific methods for performing knowledge translation, and evaluate our methods on two real-world knowledge translation tasks. As shown in the previous example, KT is useful in situations where data translation/transfer is problematic due to privacy or scalability concerns. Unlike transfer learning, which uses data in the target schema or domain to refine transferred knowledge, KT can be applied before any target data is available.

We formally define knowledge translation as the task of converting knowledge $K_{\mathcal{S}}$ in source schema $\mathcal{S}$ to equivalent knowledge $K_{\mathcal{T}}$ in target schema $\mathcal{T}$, where the correspondence between the schemas is given by some mapping $M_{\mathcal{S},\mathcal{T}}$. In general, one schema may have concepts that are more general or specific than the other, so an exact translation may not exist. We will therefore attempt to find the *best* translation, acknowledging that the best translation may still be a lossy approximation of the source knowledge.

We adopt a *probabilistic approach* to knowledge translation, in which the knowledge in the source schema, the mapping between the source and target schemas, and the resulting knowledge in the target schema are all represented as probability distributions. This gives us a consistent mathematical framework for handling uncertainty at every step in the process. This uncertainty is clearly necessary when the source knowledge is probabilistic, but it is also necessary when there is no exact mapping between the schemas, or when the correct mapping is uncertain. We propose to represent these probability distributions using *Markov random fields*, for propositional (non-relational) domains, and *Markov logic networks*, for relational domains. We will later discuss how different kinds of knowledge and mappings can be represented succinctly in these representations.

Given probability distributions for both the source knowledge and the schema mapping, we can combine them to define an implicit probability distribution in the target schema. Our goal is to find an explicit probability distribution in the target schema that is close to this implicit distribution. This gives us a compact probabilistic model that represents knowledge from the source schema as well as possible, respecting the uncertainty in both the source knowledge and the mapping. To solve this task, we propose a method for optimizing the Kullback-Leibler divergence between the two distributions. Our method is built on standard learning and inference algorithms for probabilistic graphical models.

The chapter is organized as follows. In Section 5.2., we summarize related work, such as semantic integration, distributed data mining, and transfer learning, and discuss their connections and distinctions with KT. In Section 5.3., we show how Markov random fields and Markov logic networks can represent knowledge and mappings with uncertainty. In Section 5.4., we present a variant of the

Markov logic learning algorithm to solve the problem of knowledge translation. In Section 5.5., we run experiments on synthetic and real datasets. Finally, we conclude in Section 5.6..

5.2. Related Work

We compare our knowledge translation approach with some related work, especially semantic integration, heterogeneous distributed data mining, transfer learning, and deductive knowledge translation.

5.2.1. Semantic Integration

Semantic integration aims to resolve the semantic heterogeneity between schemas or ontologies. Data integration and exchange is the mostly studied areas in semantic integration. The main task of data integration and exchange is to answer queries posed in terms of the global schema given source databases. The standard semantics of global query answering is to return the tuples in every possible database that is consistent with the global schema constraints and the mapping, i.e., the set of *certain answers*.

In the AI and Semantic Web community, researchers focus on ontologies and ontology mapping instead of databases and schema mapping. Kalfoglou and Schorlemmer (2003) review several ontology mapping frameworks and Noy (2004) discuss several formal representations of ontology mappings with different levels of expressivity for different applications of semantic integration.

Most work addressing ontology mapping to date has actually focused on *ontology matching* which defines a set of equivalence relationships between concepts and properties (e.g., IF-Map (Kalfoglou and Schorlemmer, 2002) and ONION

(Mitra and Wiederhold, 2002)). DL axioms (Arenas et al., 2012; Ritze et al., 2008) are another natural choice for representing a mapping. Some have argued that DL axioms are not sufficient for representing many useful mappings and have proposed more expressive languages. OISs (Calvanese et al., 2001) and Madhavan et al. (2002) both consider mappings as queries/views in a similar way to schema mapping. Dou et al. (2005) use first-order axioms directly to represent the mapping in a special-purpose inference engine for ontology translation. The drawback of introducing such an expressive language is that the task is not always solvable.

A main difference between data integration/exchange and knowledge translation (KT) is that KT has probabilistic semantics for the translation process, that is, it defines a distribution of possible worlds in the target schema, instead of focusing only on the tuples that are in all the possible worlds (i.e., certain answers). As a result, most work in data integration/exchange uses fragments of first-order logic with built-in predicates (e.g., comparison operators) and functions (e.g., arithmetic operators) to represent the mappings, while we can sacrifice the capability of exact inference and use first-order logic (FOL) to represent the mapping.

5.2.2. Distributed Data Mining

Efforts in distributed data mining (DDM) (see surveys in (Park and Kargupta, 2002; Caragea et al., 2005)) have made considerable progress in mining distributed data resources without putting data in a centralized location. (Caragea et al., 2005) proposes a general DDM framework with two components: one sends statistical queries to local data sources, and the other uses the returned statistics to revise the current partial hypothesis and generate further queries. For each

data mining model and algorithm (e.g., SVMs, Naive Bayes classifiers), a statistic with smaller size than the original data is used, which reduces the cost of data transmission.

Heterogeneous DDM (Caragea et al., 2005) also handles the semantic heterogeneity between the global and local schemas, in particular, those containing attributes with different granularities called Attribute Value Taxonomy (AVT). Heterogeneous DDM requires local data resources and their mappings to the global schema to translate the statistics of queries. However, KT does not require data or statistics from either the source or the target. Instead, KT uses mappings to translate the generated/mined knowledge from the source directly.

5.2.3. Transfer Learning

Transfer learning (TL) has been a successful approach to knowledge reuse Pan and Yang (2010). In traditional machine learning, only one domain and one task is involved. When the amount of data is limited, it is desirable to use data from related domains or tasks. As long as the source and target data share some similarity (e.g., in the distribution or underlying feature representation), the knowledge obtained from the source data can be used as a "prior" for the target task.

Early transfer learning work focuses on the homogeneous case in which the source and target domain have identical attributes. Recently, many other scenarios of transfer learning are studied, including heterogeneous transfer learning Yang et al. (2009), relational transfer learning Mihalkova et al. (2007); Davis and Domingos (2009), network transfer learning Fang et al. (2015); Ye et al. (2013). Some of these scenarios have similar settings as knowledge translation. For

example, heterogeneous transfer learning also deals with different representations of the data. While it uses an implicit mapping of two feature spaces (e.g., texts and images through Flickr), KT uses an explicit mapping via FOL formulas. Relational transfer learning also involves relational domains and relational knowledge. While it deals with two analogous domains (e.g., in movie and university domains, directors correspond to professors), KT focuses on a single domain with merely different representations. Moreover, relational transfer learning only handles deterministic one-to-one matchings which can be inferred with both the source and target data, while KT does not use any target data and relies on the provided explicit FOL mapping.

5.2.4. Deductive Knowledge Translation

Deductive knowledge translation (Dou et al., 2011) essentially tries to solve the same problem, but it only considers deterministic knowledge and mappings. Our KT work can handle knowledge and mappings with uncertainty, which is more general than the deterministic scenario deductive knowledge translation can handle.

See Table 5.1 for a summary of the similarities and differences between our knowledge translation (KT) approach and related work.

TABLE 5.1. Comparisons between KT and related work. We consider three aspects of a task: whether data is available, what kind of knowledge patterns are supported, and what kind of mapping is used.

	Data availability	Knowledge type	Mapping type
Data integration	Source data	Query results	GLAV mappings
Heterogeneous DDM	Source data	Propositional	AVT
Relational TL	Target data	SRL models	Matching
Ontology exchange	No data	DL axioms	Subsumption axioms
Deductive KT	No data	FOL rules	FOL
KT	No data	SRL models	FOL with uncertainty

5.3. Probabilistic Representations of Knowledge and Mappings

To translate knowledge from one schema to another, we must have a representation of the knowledge and the mappings between the two schemas. Some kinds of knowledge can be represented in propositional or first-order logic, such as hard constraints on database attribute values (e.g., date of death cannot precede date of birth) or ontology classes (e.g., Student is a subclass of Person). Some kinds of mapping can be represented logically as well, such as deterministic equivalences between attributes in the source and target schemas.

In many cases, however, knowledge and mappings are uncertain. For example, the mined source knowledge could be a probabilistic model, such as a Bayesian network. Mined knowledge in the form of predictive models, such as decision trees and association rules, is also uncertain because these models may not have perfect accuracy. Mappings between two schemas may also be uncertain, either because a perfect alignment of the concepts does not exist, or because there is uncertainty about which alignment is the best. Therefore, we propose a *probabilistic* approach to knowledge translation. In this section, we first provide background on probabilistic graphical models and then describe how they can be used to represent different types of knowledge and mappings.

5.3.1. Representation of Knowledge

Our approach to knowledge translation requires that the source and target knowledge are probability distributions represented as log-linear models. In some cases, the source knowledge mined from the data may already be represented as a log-linear model, such as a Bayesian network used for fault diagnosis or Markov

logic network modeling homophily in a social network. In other cases, we will need to convert the knowledge into this representation.

For mined knowledge represented as rules, including association rules, rule sets, and decision trees (which can be viewed as a special case of rule sets), we can construct a feature for each rule, with a weight corresponding to the confidence or probability of the rule. The rule weight has a closed-form solution based on the log odds that the rule is correct:

$$w_i = \log \frac{p(f_i)}{1 - p(f_i)} - \log \frac{u(f_i)}{1 - u(f_i)}$$

where $p(f_i)$ is the probability or confidence of the ith rule or formula and $u(f_i)$ is its probability under a uniform distribution. This allows us to convert uncertain knowledge rules into a log-linear model. This method also supports ensembles, including bagged decision trees, boosted decision trees, and random forests. Relational rules in an ontology can similarly be converted to a Markov logic network by attaching weights representing their relative strengths or confidences.

For linear classifiers, such as linear support vector machines or perceptrons, we can substitute logistic regression, a probabilistic linear classifier. The parameters of the logistic regression model can be tuned to make the classification more or less confident near the decision boundary. Neural networks can also be converted to probability distributions by introducing a random variable for each hidden unit.

Some representations are harder to represent as log-linear models. For example, kernelized SVMs and nearest neighbor classifiers do not have obvious analogs as log-linear models. Applying our method to such knowledge types might require a specialized probabilistic representation; we leave this investigation to

future work. For now, we focus on types of knowledge that are easy to represent as log-linear models, which already covers most of the common types of knowledge used in data mining.

In many cases, the knowledge we wish to translate takes the form of a conditional probability distribution, $p(\boldsymbol{Y}|\boldsymbol{X})$, or a predictive model that can be converted to a conditional probability distribution. This includes decision trees, neural networks, and other classifiers used in data mining and machine learning. The method we propose will rely on a full joint probability distribution over all variables. We can convert a conditional distribution into a joint distribution by assuming some prior distribution over the evidence, $p(\boldsymbol{X})$, such as a uniform distribution. If the source distribution $p(\boldsymbol{X}, \boldsymbol{Y})$ and target distribution $p(\boldsymbol{X}', \boldsymbol{Y}')$ are identical (and positive), then the conditional distributions $p(\boldsymbol{Y}|\boldsymbol{X})$ and $p(\boldsymbol{Y}'|\boldsymbol{X}')$ are also identical. In this case, $p(\boldsymbol{X})$ does not matter and will not affect the accuracy of the knowledge translation! In the more common case, the target distribution will be merely similar to the source distribution. To minimize the expected difference between the distributions, we need to know which evidence configurations are more likely. Thus, a good or bad choice of $p(\boldsymbol{X})$ can have some effect on the final translation quality.

5.3.2. Representation of Mappings

The relationships between heterogeneous schemas can be represented as a *mapping*. We use probabilistic models to represent mappings. Consistent with the probabilistic representation of knowledge in a database schema, the attributes are considered as random variables for non-relational domains, and the attributes or relations are considered as first-order random variables for relational domains. Let

us denote the variables in the source as $\boldsymbol{X} = \{X_1, ..., X_N\}$ and those in the target as $\boldsymbol{X}' = \{X'_1, ..., X'_M\}$. A mapping is the conditional distribution $p(\boldsymbol{X}'|\boldsymbol{X})$.

In real cases, a mapping is often represented as a set of source-to-target correspondences

$$\{p(\boldsymbol{C}'_i|\boldsymbol{C}_i), i = 1, ..., I\}$$

where $\boldsymbol{C}_i \subset \boldsymbol{X}$ and $\boldsymbol{C}'_i \subset \boldsymbol{X}'$ are sets of variables in the source and target respectively. For the credit card company example, a mapping between the two schemas may include the correspondences of "age" and "age," "salary" and "annual income," etc.

In order to obtain a global mapping between the source and target schemas using the local correspondences, we make the following two assumptions:

1. $p(\boldsymbol{C}'_i \cup \boldsymbol{C}'_j|\boldsymbol{X}) = p(\boldsymbol{C}'_i|\boldsymbol{X})p(\boldsymbol{C}'_j|\boldsymbol{X})$, or, $\boldsymbol{C}'_i \perp \boldsymbol{C}'_j|\boldsymbol{X}$, i.e., the target variable sets in the correspondences are conditionally independent given the source variables;

2. $p(\boldsymbol{C}'_i|\boldsymbol{X}) = p(\boldsymbol{C}'_i|\boldsymbol{C}_i)$, i.e., the target variable set in each correspondence conditional probability distribution is fully determined by its corresponding source variable set.

From these two assumptions, it follows that:

$$p(\boldsymbol{X}'|\boldsymbol{X}) = \prod_i p(\boldsymbol{C}'_i|\boldsymbol{X}) = \prod_i p(\boldsymbol{C}'_i|\boldsymbol{C}_i)$$

Note that these assumptions are not always correct, but they provide a good approximation of the global mapping when it is not available. This formulation also provides an easy way of representing the global mapping as a log-linear model: we

encode each correspondence as a feature using logical formulas, and then simply combine them to obtain a log-linear model representing $p(\boldsymbol{X}'|\boldsymbol{X})$. If the source schema is non-relational, the log-linear model is a Markov random field; if it is relational, the final log-linear model is a Markov logic network. This applies to both knowledge and mappings.

The weight of each formula can be estimated with the log-odds. For example, we define a probabilistic source-to-target correspondence as $q_{\mathcal{S}} \rightarrow_p q_{\mathcal{T}}$, where $q_{\mathcal{S}}$ and $q_{\mathcal{T}}$ are queries (i.e., logical formulas) of source and target schemas or ontologies, and $\rightarrow_p$ has probabilistic semantics:

$$\Pr(q_{\mathcal{T}}|q_{\mathcal{S}}) = p$$

Example 5.1 (Class correspondence)**.** If x is a graduate student, then x is a student and older than 24 with probability 0.9, and vise versa.

$$\texttt{Grad}(x) \rightarrow_{0.9} \texttt{Student}(x) \wedge \texttt{Age}(x, y) \wedge (y \geq 24)$$
$$\texttt{Grad}(x) \leftarrow_{0.9} \texttt{Student}(x) \wedge \texttt{Age}(x, y) \wedge (y \geq 24)$$

This can be converted to

$$2.2 \qquad \texttt{Grad}(x) \rightarrow (\texttt{Student}(x) \wedge \texttt{Age}(x, y) \wedge (y \geq 24))$$
$$2.2 \qquad \texttt{Grad}(x) \leftarrow (\texttt{Student}(x) \wedge \texttt{Age}(x, y) \wedge (y \geq 24))$$

Note that the second formula is a target-to-source correspondence. It is equivalent to

$$2.2 \qquad \neg\texttt{Grad}(x) \rightarrow \neg(\texttt{Student}(x) \wedge \texttt{Age}(x, y) \wedge (y \geq 24))$$

Example 5.2 (Attribute correspondence)**.** The list price of x follows a Gaussian distribution parameterized by the price of x.

$$\texttt{Price}(x, y) \rightsquigarrow \texttt{ListPrice}(x, z) \wedge (z \sim \mathcal{N}(1.1y, (0.05y)^2))$$

This is a natural extension of the above formulation to continuous attribute correspondences, which can be converted to a (hybrid) Markov logic (Wang and Domingos, 2008) formula

$$-400.0 \qquad (\texttt{Price}(x, y) \wedge \texttt{ListPrice}(x, z)) \times (z/y - 1.1)^2$$

Dong et al. (2007, 2009) proposed *probabilistic schema mappings* to handle the *uncertainty* in mappings, which is similar to our representation. They define probabilistic mapping as a triple $\mathcal{M} = (\mathcal{S}, \mathcal{T}, \Sigma)$, where Σ is a set of mapping and probability pairs

$$\{\sigma_i, \Pr(\sigma_i)\}, i = 1, \cdots, l$$

where $\sum_{i=1}^{l} Pr(\sigma_i) = 1$. Each mapping σ_i is restricted to be a one-to-one mapping, which is a set of *attribute correspondences* between $\mathcal{S}$ and $\mathcal{T}$, but it can also be extended to other types of mappings. They discussed two semantics of probabilistic mappings, namely by-table and by-tuple. In the by-table semantics, each mapping can be applied to all the data in the source. In the by-tuple semantics, multiple mappings can be applied to the subsets of tuples in the source database. Our probabilistic representation can be viewed as a compact representation of their work with the by-tuple semantics.

5.4. Knowledge Translation

In this section, we formalize the task of knowledge translation (KT) and propose a solution to this task. We have the source knowledge represented as a probabilistic model $p(\boldsymbol{X}) = p(X_1, ...X_n)$ and a probabilistic mapping $P(\boldsymbol{X}'|\boldsymbol{X})$. The probabilistic model in the target schema can be computed as

$$p(\boldsymbol{X}') = \sum_{\boldsymbol{X}} p(\boldsymbol{X}, \boldsymbol{X}') \quad \text{(Equation 5.1)}$$

$$= \sum_{\boldsymbol{X}} p(\boldsymbol{X}) p(\boldsymbol{X}'|\boldsymbol{X}) \quad \text{(Equation 5.2)}$$

$$= \sum_{\boldsymbol{X}} p(\boldsymbol{X}) \prod_i p(\boldsymbol{C}'_i|\boldsymbol{C}_i) \quad \text{(Equation 5.3)}$$

Our goal is to find a *compact* probabilistic model in the target schema (i.e., the target knowledge) without using any source variables as latent variables. This requirement is due to both efficiency (when the knowledge is being used) and understandability consideration.

We also use a log-linear model $q(\boldsymbol{X}')$ to represent this compact model. A straight-forward objective is to minimize the Kullback-Leibler divergence

$$q^* = \arg\min_q D_{\mathrm{KL}} \left[p(\boldsymbol{X}') \| q(\boldsymbol{X}') \right] \quad \text{(Equation 5.4)}$$

$$= \arg\min_q - \sum_{\boldsymbol{X}'} p(\boldsymbol{X}') \log q(\boldsymbol{X}') \quad \text{(Equation 5.5)}$$

The joint distribution $p(\boldsymbol{X}, \boldsymbol{X}')$ is also a log-linear model (see Equation Equation 5.3). The weights for a local correspondence can be computed

as:

$$\begin{aligned}\bar{\theta}(\boldsymbol{C}_i, \boldsymbol{C}'_i) &= \log p(\boldsymbol{C}'_i|\boldsymbol{C}_i) \\ &= \log \frac{\exp(\theta(\boldsymbol{C}_i, \boldsymbol{C}'_i))}{\sum_{\boldsymbol{C}'_i} \exp \theta(\boldsymbol{C}_i, \boldsymbol{C}'_i)}\end{aligned}$$

where $\theta(\boldsymbol{C}_i, \boldsymbol{C}'_i)$ are the weights of the correspondence in the probabilistic mapping model. The computation of $p(\boldsymbol{X}')$ is therefore a standard inference task of the joint model $p(\boldsymbol{X}, \boldsymbol{X}')$.

5.4.1. Parameter Learning

The parameters of the target log-linear model that minimizes
Equation Equation 5.5 can be computed via standard optimization algorithms.
A simple way to compute the objective is sampling: we first generate a sample from the source $p(\boldsymbol{X})$, and then generate a sample of $\boldsymbol{X}'$ from $p(\boldsymbol{X}'|\boldsymbol{X})$ conditioned on the source sample. In the relational domain (with Markov logic or other statistical relational models), each sample instance is a database, and we need to first decide the number of constants and create a set of ground variables with these constants.

5.4.2. Structure Learning

The structure of the target knowledge can also be learned via standard structure learning algorithms for Markov random fields or Markov logic networks. An alternative approach is to use heuristics to generate the structure first.
For deterministic one-to-one correspondences, the independences in the target schema are the same as those in the source schema up to renaming. If the correspondences are non-deterministic, we may have less independences in the

target schema, and we could have an extremely complex model with large cliques. Nonetheless, in realistic scenarios, the correspondences in a mapping are usually deterministic or nearly deterministic. Therefore, it is reasonable to pretend they are deterministic while inferring the target structure. In this way we trade off between the complexity and accuracy of the target knowledge.

First of all, for Markov logic, we use first-order cliques instead of formulas as the source structure, so that it is consistent with the propositional case. We show the pseudocode of the structure translation in Algorithm 1. It is considered as a structure learning process. The first step (Line 1-8) is to remove the variables that do not have a correspondence in the target schema. This can be done by standard variable elimination (Koller and Friedman, 2009; Poole, 2003) without calculating parameters. However, exact variable elimination may create very large cliques and be very expensive, especially in Markov logic in the relational domains. Therefore, we approximate it by only merging two cliques at a time. For relational case, the merging involves a first-order unification operation (Russell and Norvig, 2003; Poole, 2003). When multiple most general unifiers exist, we simply include all the resulting new cliques. In the second step (Line 9-15), we replace each variable with the corresponding variables in the target schema. This also involves first-order unification in the relational case. If there are many-to-many correspondences, we may generate multiple target cliques from one source clique.

Example 5.3. Given the source Markov logic:

$$\texttt{Grad}(x) \rightarrow \texttt{AgeOver25}(x)$$

$$\texttt{AgeOver25}(x) \rightarrow \texttt{GoodCredit}(x)$$

Algorithm 1 Structure Translation (MRFs or MLNs)

Input: The source schema $\mathcal{S}$, source structure (propositional or first-order cliques) $\Phi = \{\phi_i\}$, and mapping $\mathcal{M}$.
Output: The target structure $\Phi'_{\mathcal{M}}$.

1: **for each** variable (or first-order predicate) $P \in \mathcal{S}$ that does not appear in $\mathcal{M}$ **do**
2: Let Φ_P denote all the cliques containing P
3: Remove Φ_P from Φ
4: **for each** pair of cliques in Φ_P **do**
5: Merge the two cliques and remove P
6: Insert the resulting clique back to Φ
7: **end for**
8: **end for**
9: **for each** clique $\phi \in \Phi$ **do**
10: **for each** variable P in ϕ **do**
11: Let $P'_{\mathcal{M}}$ be all possible correspondences of P
12: **end for**
13: Let $\phi'_{\mathcal{M}}$ denote all possible correspondences of ϕ: $\phi'_{\mathcal{M}} \leftarrow$ Cartesian product of $P'_{\mathcal{M}}$
14: Add $\phi'_{\mathcal{M}}$ to $\Phi'_{\mathcal{M}}$
15: **end for**

and the mapping:

$$2.2 \qquad \texttt{Grad}(x) \vee \texttt{Undergrad}(x) \leftrightarrow \texttt{Student}(x)$$

$$3.0 \qquad \texttt{GoodCredit}(x) \leftrightarrow \texttt{HighCreditScore}(x)$$

We first eliminate $\texttt{AgeOver25}(x)$ from the source structure because it does not occur in the mapping, and we get a new clique

$$\{\texttt{Grad}(x), \texttt{GoodCredit}(x)\}$$

Then we translate the clique based on the mapping, which gives

$$\{\texttt{Student}(x), \texttt{HighCreditScore}(x)\}$$

5.5. Experiments

To evaluate our methods, we created two knowledge translation tasks: one on a non-relational domain (NBA) and one on a relational domain (University). In each knowledge translation task, we have 2 different database schemas as the source and target schemas and a dataset for each schema. The input of a knowledge translation system is the source knowledge and the mapping between the source and target schema. The output of a knowledge translation system is the target knowledge (i.e., a probabilistic model in terms of the target schema).

We obtained the source knowledge (i.e., a probabilistic model in the source) by performing standard learning algorithms on the source datasets, and created the probabilistic schema mappings manually. Our approach can potentially support automatically discovered mappings (e.g., Rahm and Bernstein (2001)) as well,

but we use manually created mappings in the experiments for two reasons: first, our method strongly relies on the quality of the mapping, so we want to use more accurate mappings for a quantitative analysis of the method itself; second, we use schemas with plenty of semantic heterogeneity to make the translation problem non-trivial, which is a difficult scenario for automatic tools.

5.5.1. Methods and Baselines

See Table 5.2 for an overview of the methods and baselines we compare in our experiments. Below, we describe them in more detail.

We evaluate four different versions of our proposed probabilistic knowledge translation approach described in the previous section. All of them use the source knowledge base and probabilistic mapping to generate a sampled approximation of the distribution in the target schema, and all of them use these samples to learn an explicit distribution in the target schema. The difference between them is their approach to knowledge structure. **LS-**K_S ("learned structure") learns the structure directly from the samples, which is the most flexible approach. **TS-**K_S ("translated structure") uses a heuristic translation of the structure from the source knowledge base, which may help avoid overfitting or underfitting when only a small number of samples is used to approximate the translated distribution. Since structure learning is often more challenging than weight learning, **TS-**K_S also avoids a potentially computationally intensive process. **ES-**K_S ("empty structure") is a simple baseline in which the target knowledge base is limited to a marginal distribution. This shows what can be achieved without any structure at all.

We also compare to several baselines that make use of additional data. When there is data D_S in the source schema, we can use the probabilistic mapping to

translate it to the target schema and learn models from the translated source data. **LS-**D_S and **MS-**D_S learn models from translated source data, using learned and manually specified structures, respectively. (The manually specified structures are necessary for the relational domain because MLN structure learning did not work well.) When there is data D_T in the target schema, we can learn from this data directly. **LS-**D_T learns models from target data with learned and manually specified structures, respectively. These methods represent an unrealistic "best case" since they use data that is typically unavailable in knowledge translation tasks.

TABLE 5.2. Overview of the different methods used in our experiments.

KT methods using source knowledge (K_S)	
LS-K_S	Learn struct. and weights from K_S
TS-K_S	Translate struct., learn weights from K_S
ES-K_S	Empty struct., learn marginals from K_S
Baselines using translated source data (D_S)	
LS-D_S	Learn struct. and weights from D_S
MS-D_S	Manually specify struct., learn weights from D_S
Baselines using additional target data (D_T)	
LS-D_T	Learn struct. and weights D_T

5.5.2. Evaluation Criteria

We evaluate our knowledge translation methods according to two criteria:

First, *how well does the translated knowledge represent the target domain*? This is the problem we wish to solve with KT – obtain an accurate model in the target domain without seeing any data. We measure this using the pseudo-log-likelihood (PLL) of held-out target data. When comparing two models, the one with the higher PLL better captures the target distribution.

Second, *how well does the translated knowledge represent the source distribution and mapping?* Since this is what our KT methods are designed to do, we want to know how well they do it. We measure this using the PLL of the translated knowledge on translated held-out source data. When comparing two models of translated knowledge, the one that more accurately captures the relationships from the source domain should have higher PLL on data translated from the source domain. The advantage of this second measure is that it controls for differences between the source and target distributions. If the source and target distributions are significantly different, then a more accurate translation of the source knowledge could lead to a less accurate distribution in the target domain.

For relational domains, we use weighted pseudo-log-likelihood (WPLL), where for each predicate r, the PLL of each of its groundings is weighted by the $c_r = 1/g_r$, where g_r is the number of its groundings.

5.5.3. Non-Relational Domain (NBA)

We collected information on basketball players in the National Basketball Association (NBA) from two websites, the NBA official website `nba` (as the source schema) and the Yahoo NBA website `yahoo` (as the target schema). The schemas of these two datasets both have the name, height, weight, position and team of each player. In `nba`, there are 3 values for a player's position: forward, guard and center, while in `yahoo`, a finer taxonomy of 7 values is used: forward, small, power, guard, point, shooting and center.

We modified the original schemas in order to make the mapping between them more interesting (and challenging). In our modified `nba` and `yahoo` schemas, the height and weight are in imperial and metric units respectively. Also, in `nba`,

we discretize height and weight into 5 equal-width ranges. In `yahoo`, we discretize them into 5 equal-frequency ranges.

We used the Libra Toolkit[1] (Lowd and Rooshenas, 2015) for creating the source knowledge and for performing the learning and inference subroutines required by the different knowledge translation approaches.

We first left out 1/5 of the data instances in the source and target dataset as the testing sets. For the remaining source dataset, we used the decision tree structure learning (DTSL) (Lowd and Davis, 2014) to learn the source knowledge. We used standard 4-fold cross validation to determine the parameters of the learning algorithm. The parameters include κ, prior, and mincount for decision tree learning, and l_2 for weight learning.

The mapping is also represented as a Markov random field. For the numerical attributes (e.g., weight and height), the correspondence is originally a unit conversion formula

$$h' = h \times 39.3701$$

After we discretize these attributes, we can calculate the correspondence distribution of the ranges by making a simple assumption that each value range is uniformly distributed, e.g.,

$$p(h' \in (-\inf, 73.5] | h \in (1.858, 1.966]) = 0.082$$
$$p(h' \in (73.5, 76.5] | h \in (1.858, 1.966]) = 0.706$$
$$p(h' \in (76.5, 78.5] | h \in (1.858, 1.966]) = 0.212$$
$$p(h' \in (78.5, +\inf] | h \in (1.858, 1.966]) = 0$$

[1] `http://libra.cs.uoregon.edu/`

These conditional distributions are then converted to weights of the Markov random field representing the mapping.

We use Gibbs sampling implemented in Libra for the sampling algorithm in the knowledge translation approaches. For LS-K_S and TS-K_S, we draw N samples from the source knowledge probability distribution. To avoid correlations between the samples, each sample is generated from an independent Gibbs sampling chain with 1000 burn-in iterations. We then use the probabilistic mapping to draw 1 target sample for each source sample. For LS-D_S, suppose we have N_S instances in the source dataset. We use the probabilistic mapping to draw N/N_S target samples for each source instance, such that the total number of target instances is also N.

LS-K_S and TS-K_S both perform weight learning with an l_2 prior. For structure translation with TS-K_S, we only translate features for which the absolute value of the weight is greater than a threshold θ. These two parameters are tuned with cross validation over a partition of the samples.

5.5.3.1. Results

See Figures 5.1 for learning curves comparing our methods to the baselines. Along the x axis, we vary the number of samples N used to approximate the implicit distribution. Intuitively, performance increases with more samples but eventually plateaus.

From Figure 5.1, we see that translated knowledge (LS-K_S and TS-K_S) is more accurate on the target data than knowledge learned from translated source data (LS-D_S). This confirms that *KT can be as accurate as data translation*, but with the advantage of not requiring any data. Note that the performance of LS-D_S degrades as the number of samples increases. This is because the source dataset

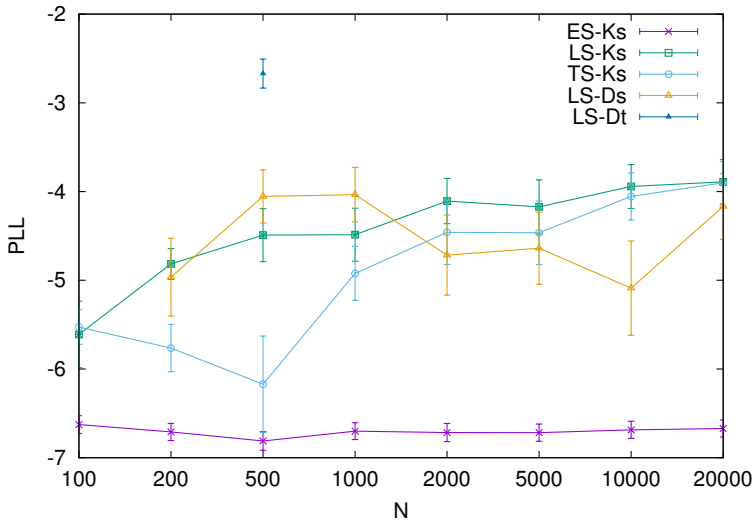

FIGURE 5.1. PLL for KT methods and baselines on target data in the NBA domain

is small. When the source data are translated to the target schema, we generated many samples for each data instance, and the resulting translated dataset would be a mixture model of the translation of each data instance, which is different from the real distribution. Finally, as expected, the model learned directly on the target data (LS-D_T) has the best PLL on the target data, since it could observe the target distribution directly.

In Figure 5.2, we see that translated knowledge (LS-K_S and TS-K_S) is almost as accurate on translated source data as models trained directly on translated source data. Therefore, our methods are performing close to optimally. For this domain, we do not see a large difference between learning the structure (LS-K_S) and heuristically translating the structure (TS-K_S).

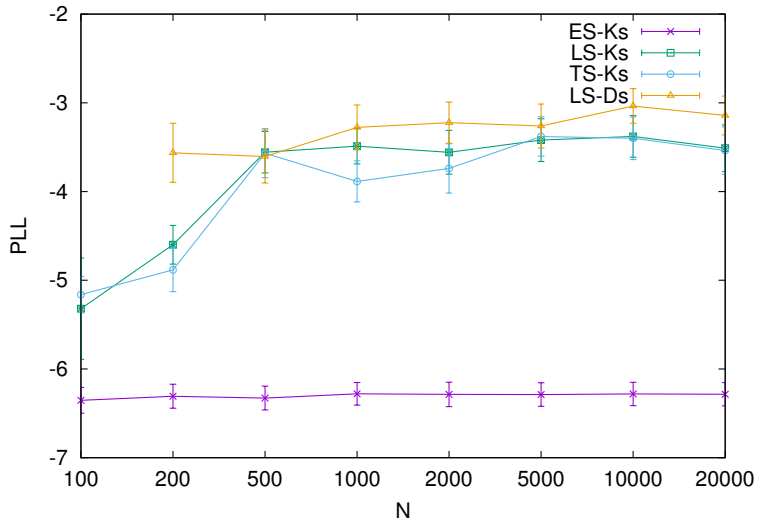

FIGURE 5.2. PLL for KT methods and baselines on translated source data in the NBA domain

5.5.4. Relational Domain (University)

We use the `UW-CSE` dataset[2] (Domingos and Lowd, 2009) and the `UO-CIS` dataset which we collected from the Computer and Information Science Department of the University of Oregon. The `UW-CSE` dataset was introduced by Richardson and Domingos (2006) and is widely used in statistical relational learning research. In this University domain, we have concepts such as persons, courses, and publications; attributes such as PhD student stage and course level; and relations such as advise, teach, and author. The schemas of the two databases differ in their granularities of concepts and attribute values. For example, in `UW-CSE`, the professors have positions (represented as attributes) such as "faculty," "affiliate," and "adjunct," while in `UO-CIS`, we have concepts such as "professor," "associate professor," and "assistant professor." Also, `UW-CSE` graduate courses

[2] `http://alchemy.cs.washington.edu/data/uw-cse/`

are marked as level 500, while UO-CIS has both graduate courses at level 600 and combined undegraduate/graduate courses at level 4/500.

Our methods in this relational domain are similar to those in the non-relational domain. We use Alchemy [3] for learning and inference in Markov logic networks. We obtain the source knowledge by manually creating the formulas in the source schema and then using the source data to learn the weights.

We use MC-SAT (Poon and Domingos, 2006) as the sampling algorithm for these experiments. As we discussed in the previous section, we first need to decide the number of constants for each type in the domain. Since the behavior of a Markov logic network is highly sensitive to the number of constants, we want to keep the number of constants similar to the original dataset from which the model is learned. In the experiments, we set the number of constants of each type to be the average number over all training databases, multiplied by a scalar λ. Larger values of λ increase the number of possible relationships among the objects, while smaller values of λ make the inference procedure more efficient. We set $\lambda = 1/2$ in our experiments. We also draw N samples from the source distribution and 1 target sample from each source sample and the mapping distribution. Here N does not have to be large, because each sample instance of a relational domain is itself a database. We set N to 1, 2 and 5 in our experiments.

We set the l_2 prior for weight learning to 10, based on cross-validation over samples. The results are shown in Table 5.3.

[3] http://alchemy.cs.washington.edu/alchemy1.html

5.5.4.1. Results

See Tables 5.3 and 5.3 for results on target data and translated source data, respectively. Learning MLN structure did not work well in these domains, so we use manually specified structures when learning from translated source data (MS-D_S). From a single sample, the translated source data and manually specified structure (MS-D_S) were more effective than knowledge translation with translated structure (TS-K_S). However, as we increase the number of samples, the performance of TS-K_S improves substantially. With 5 samples, the performance of TS-K_S becomes competitive with that of MS-D_S, again demonstrating that knowledge translation can achieve comparable results to data translation but without data. When evaluated on translated source data, TS-K_S shows the same trend of improving with the number of samples, but its performance with 5 relational samples is slightly worse than MS-D_S.

TABLE 5.3. Evaluation on the target dataset for the university domain.

Method	WPLL on target			WPLL on source		
# Samples	1	2	5	1	2	5
ES-K_S	-3.77	-3.76	-3.83	-3.54	-3.44	-3.39
LS-K_S	-12.07	-3.82	-3.48	-9.19	-3.72	-1.51
TS-K_S	-2.51	-2.80	-1.79	-2.05	-2.10	-0.97
LS-D_S	-3.70	-3.01	N/A	-1.23	-1.23	N/A
MS-D_S	-1.94	-1.91	-1.76	-1.22	-0.93	-0.61
LS-D_T	-1.33					
MS-D_T	-1.18					

5.6. Summary

Knowledge translation is an important task towards knowledge reuse where the knowledge in the source schema needs to be translated to a semantically heterogeneous target schema. Different from data integration and transfer learning,

knowledge translation focuses on the scenario that the data may not be available in both the source and target. We propose a novel probabilistic approach for knowledge translation by combining probabilistic graphical models with schema mappings. We have implemented an experimental knowledge translation system and evaluated it on two real datasets for different prediction tasks. The results and comparison with baselines show that our approach can obtain comparable accuracy without data.

CHAPTER VI

CONCLUSION

We investigate three problems related to knowledge management and manipulation. First, given an automatically extracted knowledge base, how to leverage the dependencies among the KB instances with different degrees of confidence to obtain a refined KB with better accuracy. Second, how to improve ontology matching by utilizing the aligned knowledge rules contained in the two ontologies. Third, when we apply our knowledge to a target task using a different schema, how to effectively translate the knowledge given the mapping between the source and target schemas.

We propose to use Markov logic networks (MLNs), a powerful statistical relational learning model that enhances first-order logic with uncertainty, to deal with these problems. MLNs are flexible and expressive in representing domain knowledge, knowledge rules, knowledge bases, and schema mappings, as well as in representing the uncertainty often presented in them. We propose MLN formulations for these problems, and use standard MLN learning and inference algorithms with certain adaptations to solve these problems.

The main contributions of the dissertation are summarized as follows:

1. We introduce a novel paradigm by which ontology-based information extraction systems can be improved, in terms of the quality and quantity of the knowledge it creates. Information from several sources, including the generated knowledge base with confidence values for each instance and the ontological constraints, is perfectly integrated in a unified framework with

Markov logic. We also propose adaptations of the existing Markov logic algorithms to fit the extremely large size of a typical knowledge base.

2. We show how to use knowledge rules representing common types of domain models to obtain more accurate alignments of two ontologies. Our approach is especially effective in identifying the correspondences of numerical and nominal datatype properties. By incorporating complex concepts, our approach is also capable of discovering complex correspondences, which is a very difficult scenario in the ontology matching task.

3. We formally define the problem of knowledge translation (KT), which allows knowledge with different representations to be reused, even when data is unavailable. We propose a novel *probabilistic* approach for KT by combining probabilistic graphical models with schema mappings.

The experiments validate the effectiveness of the MLN-based methods for these problems.

6.1. Future Work

6.1.1. Extending Markov Logic Networks

Markov logic is a powerful, expressive and easily interpretable language of statistical relational learning. However, its representation power is still limited in some cases. An MLN can be viewed as a ground Markov random field with tied weights for similar features over different parts of the network (i.e., different instantiations of the same feature template), and therefore the number of parameters is usually much smaller than that of a MRF. As a result, An MLN tends to underfit because the size of data is usually much larger than the number

of features. Also, the additivity of feature weights is sometimes not desirable especially in complex networks. For instance, in a social network, we can use a formula

$$w_1 \quad \texttt{Friend}(x, y) \rightarrow \texttt{Popular}(x),$$

to indicate that a person with many friends is likely to be a popular person. This MLN not only oversimplifies this relation, but is also not accurate because intuitively the popularity of a person should not increase linearly with the number of his/her friends.

To overcome this problem, we can use more features. For the above example, we can add another formula with the existential quantifier:

$$w_2 \quad (\exists \texttt{Friend}(x, y)) \rightarrow \texttt{Popular}(x)$$

The original definition of MLNs only supports first-order clauses as features. It is straightforward to extends MLNs to incorporate arbitrary FOL formulas, including the ones with existential quantification. However, the grounding of existential quantification is extremely large (proportional to the domain size) and therefore not scalable in practice.

There has been other efforts in extending the log-linear model of MLNs with more powerful features other than FOL formulas. Recursive Random Fields (RRFs) (Lowd and Domingos, 2007b) use the probabilities defined by a nested Markov logic networks as the features. Expential Random Graph Models (ERGMs) (Hunter and Handcock, 2006) define a curved exponential family model that uses nonlinear parameters to represent structural properties of complex networks. These attempts may be further extended.

A related issue with MLNs is that their modeling performance degrades when the domain size changes. Notice that the above formula will effectively adds a prior to the marginal distribution of `Friend` and `Popular`. If the domain size keeps unchanged, we could use atomic formulas with certain weights to offset the influence from this formula and get the marginal distributions that we want. However, when the number of people in the domain varies, the marginal distributions of `Friend` and `Popular` will change accordingly, in a way that we would not expect.

Jain et al. (2007) noticed this problem and proposed to adjust the weights when the domain size changes so that the marginal distributions keep the same. However, sometimes we would expect the marginal distributions of some predicates to change. For instance, we would expect the marginal distribution of `Friend` decreases as the social network becomes larger.

6.1.2. Scalability of Markov Logic Networks

The ground network of an MLN is usually very large. For example, in Chapter III, we have to use a heuristic method to restrict the size of the ground network. With the help of parallel and distributed computing, we could make MLN more scalable in a systematic way.

The difficulty of designing of a parallel computing system depends on the task and algorithm. The parallel machine learning methods can be roughly categorized into data parallelism, model parallelism, and other types of parallelism. In data parallelism, a large dataset is partitioned into batches and used for computation in parallel. In model parallelism, the machine learning model (e.g., a graph or

network) is decomposed into parts and the computations are done for each part in parallel. Other parallelism includes running multiple Markov chains in parallel.

There has been general frameworks and APIs for parallel and distributed machine learning, e.g., GraphLab (a.k.a. Dato), Spark. Some of these frameworks are specific for graph-based algorithms (e.g., GraphLab, Spark GraphX) by leveraging the graph partitioning algorithms for model parallelisms. Also, people have developed parallel systems for specific tasks and algorithms (e.g., Loopy belief propagation (Gonzalez et al., 2009)).

Currently, to the best knowledge of the author, there is no distributed implementation of MLNs yet. Such systems can be developed with the distributed machine learning toolkits, to make the learning and inference of MLNs more scalable.

6.1.3. Knowledge Base Refinement

In the experiments for knowledge base refinement task, we recruited an undergraduate student to label the data for training and testing. For a general purpose knowledge base, such human labeling is usually slow, expensive, and most importantly, error prone. We can use crowdsourcing to get the lables more easily, economically and with higher quality. More training data means we can add more features, for instance, different priors for different extractor components in NELL. An even more efficient method is active learning, in which learning algorithms can actively query the labels.

We would also like to explore doing unsupervised or semi-supervised learning, to automatically learn the strength of these relationships without requiring many human labels.

In the current method, we incorporate patterns with a standalone logistic regression model. It would be ideal to integrate this model into the MLN. However, it will dramatically increase the size of the ground network. In the future, if we have a more scalable implementation of MLN, this method can be studied.

6.1.4. Knowledge Translation

Log-linear models, such as Markov random fields and Markov logic networks, already cover most of common types of knowledge used in data mining. In the future work, we will extend our approach to the knowledge types which are harder to represent as log-linear models, such as SVMs, nearest neighbor classifiers, and neural networks. It might require a specialized probabilistic representation.

We would also like to study more on translating discriminative models $p(\boldsymbol{Y}|\boldsymbol{X})$. If we make similar independence assumptions as described in Chapter V, we have

$$p(\boldsymbol{Y}'|\boldsymbol{X}') = p(\boldsymbol{Y}'|\boldsymbol{Y})p(\boldsymbol{Y}|\boldsymbol{X})p(\boldsymbol{X}|\boldsymbol{X}')$$

Note that here $p(\boldsymbol{X}|\boldsymbol{X}')$ is the mapping defined in the reverse direction. Similar to the case of generative models, we want a compact model $q(\boldsymbol{Y}'|\boldsymbol{X}')$ that is close to $p(\boldsymbol{Y}'|\boldsymbol{X}')$. A straightforward method is to make an estimation of $p(\boldsymbol{X}')$, and compute the target distribution $p(\boldsymbol{X}', \boldsymbol{Y}')$, from which a discriminative model can be learned. Alternatively, if given the forward mapping $p(\boldsymbol{X}'|\boldsymbol{X})$, we can also make an estimation of $p(\boldsymbol{X})$ and follow the same procedure. Obviously, the estimation of $p(\boldsymbol{X})$ or $p(\boldsymbol{X}')$ has a great impact to the distance of the true distribution $p(\boldsymbol{Y}'|\boldsymbol{X}')$ and the compact model $q(\boldsymbol{Y}'|\boldsymbol{X}')$.

For larger domains and knowledge related to hundreds or thousands of concepts, manually specified probabilistic mappings seem unpractical. Our method

to knowledge translation supports automatically discovered mappings as well. Moreover, it naturally supports probabilistic output of an automatic schema mapping system. As automatic tools are not yet reliable, and the performance of our method greatly depends on the mapping, we can also combine the automatic tools with expert efforts to balance between the quality and cost.

Finally, a related task of knowledge integration (KI) can be studied in the future. In KI, we have knowledge from multiple sources with different schemas, and we would like to integrate the knowledge and represent it in a different target schema. KI is a natural extension of KT, and it is more difficult. In KI, many scenarios and challenges can be considered, e.g., how to represent the mappings of multiple schemas, how to integrate different forms of source knowledge, etc.

6.2. Concluding Remarks

This dissertation presents three problems in knowledge representation, management and manipulation, and proposes solutions based on Markov logic. This suggests great potential of using Markov logic in knowledge management related research.

Logic and probability theories have been two major branches of AI research. First-order logic and description logic have been the standard theme of traditional AI since the early years, and knowledge-based systems are one of the most important applications. More recently, statistical and machine learning methods, such as graphical models, SVMs, and neural networks, become hot topics of AI. Markov logic is one of the successful methods intented to combine these branches and provide a universal solution for the theoretical foundation of AI.

The lessons we learned from this dissertation are threefold. First, FOLs are good representations for various types of knowledge. We used different kinds of FOL encodings of knowledge for different tasks. Second, uncertainty is a desirable feature of knowledge. Currently, the area of knowledge representation and Semantic Web mostly focus on deterministic methods such as description logic and its variants. It would be much more practical to consider probabilistic models such as Markov logic as the future direction. Third, compared with heuristic methods, such as the knowledge integrator (KI) in NELL, and the systems that combine several strategies for ontology matching, probabilistic methods are often not only more coherent and elegant, but also perform better.

Yet, we still face great challenges when applying Markov logic to knowledge management tasks, as well as many other applications. The breakthrough of AI call for a universal framework that is more expressive, more scalable, and combines the advantages of both the logical and probabilistic approaches.

REFERENCES CITED

Arenas, M., Botoeva, E., Calvanese, D., Ryzhikov, V., and Sherkhonov, E. (2012). Representability in DL-Lite$_R$ knowledge base exchange. In Kazakov, Y., Lembo, D., and Wolter, F., editors, *Description Logics*, volume 846 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Banko, M., Cafarella, M. J., Soderland, S., Broadhead, M., and Etzioni, O. (2007). Open information extraction from the web. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence*, IJCAI'07, pages 2670–2676, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Bröcheler, M., Mihalkova, L., and Getoor, L. (2010). Probabilistic similarity logic. In Grünwald, P. and Spirtes, P., editors, *UAI 2010, Proceedings of the Twenty-Sixth Conference on Uncertainty in Artificial Intelligence, Catalina Island, CA, USA, July 8-11, 2010*, pages 73–82. AUAI Press.

Buitelaar, P. and Siegel, M. (2006). Ontology-based information extraction with SOBA. In *LREC*, pages 2321–2324.

Calvanese, D., De Giacomo, G., and Lenzerini, M. (2001). Ontology of integration and integration of ontologies. In *Proceedings of the 2001 Description Logic Workshop (DL 2001)*, pages 10–19. CEUR Electronic Workshop Proceedings.

Caragea, D., Zhang, J., Bao, J., Pathak, J., and Honavar, V. (2005). Algorithms and software for collaborative discovery from autonomous, semantically heterogeneous, distributed information sources. In *Proceedings of the 16th International Conference on Algorithmic Learning Theory*, ALT'05, pages 13–44, Berlin, Heidelberg. Springer-Verlag.

Carlson, A., Betteridge, J., Hruschka, Jr., E. R., and Mitchell, T. M. (2009). Coupling semi-supervised learning of categories and relations. In *Proceedings of the NAACL HLT 2009 Workskop on Semi-supervised Learning for Natural Language Processing*.

Carlson, A., Betteridge, J., Kisiel, B., Settles, B., Hruschka, Jr., E. R., and Mitchell, T. M. (2010a). Toward an architecture for never-ending language learning. In *Proceedings of the Twenty-Fourth Conference on Artificial Intelligence (AAAI 2010)*.

Carlson, A., Betteridge, J., Wang, R. C., Hruschka, Jr., E. R., and Mitchell, T. M. (2010b). Coupled semi-supervised learning for information extraction. In *Proceedings of the Third ACM International Conference on Web Search and Data Mining (WSDM 2010)*.

Cimiano, P., Handschuh, S., and Staab, S. (2004). Towards the self-annotating web. In *WWW*, pages 462–471.

Cimiano, P., Ladwig, G., and Staab, S. (2005). Gimme' the context: context-driven automatic semantic annotation with C-PANKOW. In *WWW*, pages 332–341.

Cotterell, M. E. and Medina, T. (2013). A Markov model for ontology alignment. *CoRR*.

Craven, M., Dipasquo, D., Freitag, D., Mccallum, A., Mitchell, T., and Nigam, K. (1999). Learning to construct knowledge bases from the world wide web. *Artificial Intelligence*, 118:69–113.

Davis, J. and Domingos, P. (2009). Deep transfer via second-order Markov logic. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 217–224, New York, NY, USA. ACM.

Dhamankar, R., Lee, Y., Doan, A., Halevy, A., and Domingos, P. (2004). iMAP: Discovering complex semantic matches between database schemas. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, pages 383–394.

Doan, A., Madhavan, J., Domingos, P., and Halevy, A. (2002). Learning to map between ontologies on the Semantic Web. In *Proceedings of the 11th international conference on World Wide Web*, pages 662–673.

Doan, A., Madhavan, J., Domingos, P., and Halevy, A. (2004). Ontology matching: A machine learning approach. In Staab, S. and Studer, R., editors, *Handbook on Ontologies in Information Systems*, pages 385–403. Springer, New York, NY, USA.

Domingos, P. and Lowd, D. (2009). *Markov Logic: An Interface Layer for Artificial Intelligence*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool.

Dong, X., Halevy, A. Y., and Yu, C. (2007). Data integration with uncertainty. In *VLDB '07: Proceedings of the 33rd International Conference on Very Large Data Bases*, pages 687–698. VLDB Endowment.

Dong, X. L., Halevy, A., and Yu, C. (2009). Data integration with uncertainty. *The VLDB Journal*, 18(2):469–500.

Dou, D., McDermott, D., and Qi, P. (2005). Ontology translation on the semantic web. In Spaccapietra, S., Bertino, E., Jajodia, S., King, R., and McLeod, D., editors, *Journal on Data Semantics II*, pages 35–57. Springer-Verlag, Berlin, Heidelberg.

Dou, D., Qin, H., and Liu, H. (2011). Semantic translation for rule-based knowledge in data mining. In *Proceedings of the 22nd International Conference on Database and Expert Systems Applications*, volume Part II of *DEXA'11*, pages 74–89, Berlin, Heidelberg. Springer-Verlag.

Etzioni, O., Cafarella, M., Downey, D., Popescu, A.-M., Shaked, T., Soderland, S., Weld, D. S., and Yates, A. (2005). Unsupervised named-entity extraction from the web: An experimental study. *Artif. Intell.*, 165(1):91–134.

Euzenat, J. and Shvaiko, P. (2007). *Ontology Matching*. Springer-Verlag New York, Inc., Secaucus, NJ, USA.

Fang, M., Yin, J., Zhu, X., and Zhang, C. (2015). TrGraph: Cross-network transfer learning via common signature subgraphs. *IEEE Trans. Knowl. Data Eng.*, 27(9):2536–2549.

Getoor, L. and Taskar, B., editors (2007). *Introduction to Statistical Relational Learning*. Adaptive Computation and Machine Learning. The MIT Press.

Gogate, V. and Dechter, R. (2011). Samplesearch: Importance sampling in presence of determinism. *Artif. Intell.*, 175(2):694–729.

Gonzalez, J., Low, Y., and Guestrin, C. (2009). Residual splash for optimally parallelizing belief propagation. In Dyk, D. A. V. and Welling, M., editors, *AISTATS*, volume 5 of *JMLR Proceedings*, pages 177–184. JMLR.org.

Hu, W., Chen, J., Zhang, H., and Qu, Y. (2011). Learning complex mappings between ontologies. In *Proceedings of Joint International Semantic Technology Conference*, pages 350–357.

Huber, J., Sztyler, T., Noessner, J., and Meilicke, C. (2011). CODI: Combinatorial optimization for data integration–results for OAEI 2011. *Ontology Matching*, page 134.

Hunter, D. R. and Handcock, M. S. (2006). Inference in curved exponential family models for networks. *Journal of Computational and Graphical Statistics*, 15:565–583.

Huynh, T. and Mooney, R. (2008). Discriminative structure and parameter learning for Markov logic networks. In *Proceedings of the 25th International Conference on Machine Learning*, pages 416–423. ACM.

Jain, D., Kirchlechner, B., and Beetz, M. (2007). Extending markov logic to model probability distributions in relational domains. In Hertzberg, J., Beetz, M., and Englert, R., editors, *KI 2007: Advances in Artificial Intelligence*, volume 4667 of *Lecture Notes in Computer Science*, pages 129–143. Springer Berlin Heidelberg.

Jiménez-Ruiz, E., Grau, B. C., and Zhou, Y. (2012). LogMap 2.0: Towards logic-based, scalable and interactive ontology matching. In *Proceedings of the 4th International Workshop on Semantic Web Applications and Tools for the Life Sciences*, SWAT4LS '11, pages 45–46.

Kalfoglou, Y. and Schorlemmer, M. (2002). Information-flow-based ontology mapping. In *In Proceedings of the 1st International Conference on Ontologies, Databases and Application of Semantics*, pages 1132–1151. Springer.

Kalfoglou, Y. and Schorlemmer, M. (2003). Ontology mapping: The state of the art. *Knowledge Engineering Review*, 18(1):1–31.

Kietz, J., Maedche, A., and Volz, R. (2000). A method for semi-automatic ontology acquisition from a corporate intranet. *EKAW-2000 Workshop "Ontologies and Text"*.

Kindermann, R. and Snell, J. L. (1980). *Markov random fields and their applications*, volume 1. American Mathematical Society Providence, RI.

Koller, D. and Friedman, N. (2009). *Probabilistic Graphical Models: Principles and Techniques*. MIT Press.

Lao, N., Mitchell, T., and Cohen, W. W. (2011). Random walk inference and learning in a large scale knowledge base. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 529–539, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Lenzerini, M. (2002). Data integration: A theoretical perspective. In *Proceedings of the Twenty-first ACM SIGMOD-SIGACT-SIGART Symposium on Principles of Database Systems*, PODS '02, pages 233–246, New York, NY, USA. ACM.

Levenshtein, V. (1966). Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10:707.

Lowd, D. and Davis, J. (2014). Improving markov network structure learning using decision trees. *Journal of Machine Learning Research*, 15(1):501–532.

Lowd, D. and Domingos, P. (2007a). Efficient weight learning for Markov logic networks. In *In Proceedings of the Eleventh European Conference on Principles and Practice of Knowledge Discovery in Databases*, pages 200–211.

Lowd, D. and Domingos, P. (2007b). Recursive random fields. In Veloso, M. M., editor, *IJCAI*, pages 950–955.

Lowd, D. and Rooshenas, A. (2015). The libra toolkit for probabilistic models. *arXiv preprint arXiv:1504.00110*.

Madhavan, J., Bernstein, P. A., Domingos, P., and Halevy, A. Y. (2002). Representing and reasoning about mappings between domain models. In Dechter, R. and Sutton, R. S., editors, *AAAI/IAAI*, pages 80–86. AAAI Press / The MIT Press.

Mao, M., Peng, Y., and Spring, M. (2010). An adaptive ontology mapping approach with neural network based constraint satisfaction. *Web Semantics*, 8(1):14–25.

Maynard, D. (2006). Metrics for evaluation of ontology-based information extraction. In *In WWW 2006 Workshop on Evaluation of Ontologies for the Web*.

McDowell, L. and Cafarella, M. J. (2006). Ontology-driven information extraction with ontosyphon. In *International Semantic Web Conference*, pages 428–444.

Melnik, S., Garcia-Molina, H., and Rahm, E. (2002). Similarity flooding: A versatile graph matching algorithm. In *Proceedings of Eighteenth International Conference on Data Engineering*.

Mihalkova, L., Huynh, T., and Mooney, R. J. (2007). Mapping and revising Markov logic networks for transfer learning. In *Proceedings of the 22nd National Conference on Artificial Intelligence*, volume 1 of *AAAI'07*, pages 608–614. AAAI Press.

Mitchell, T., Cohen, W., Hruschka, E., Talukdar, P., Betteridge, J., Carlson, A., Dalvi, B., Gardner, M., Kisiel, B., Krishnamurthy, J., Lao, N., Mazaitis, K., Mohamed, T., Nakashole, N., Platanios, E., Ritter, A., Samadi, M., Settles, B., Wang, R., Wijaya, D., Gupta, A., Chen, X., Saparov, A., Greaves, M., and Welling, J. (2015). Never-ending learning. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence (AAAI-15)*.

Mitra, P. and Wiederhold, G. (2002). Resolving terminological heterogeneity in ontologies. In *Workshop on Ontologies and Semantic Interoperability at the 15th European Conference on Artificial Intelligence (ECAI-2002)*.

Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective*. The MIT Press.

Nath, A. and Domingos, P. (2010). Efficient belief propagation for utility maximization and repeated inference. In *AAAI*.

Niepert, M., Meilicke, C., and Stuckenschmidt, H. (2010). A probabilistic-logical framework for ontology matching. In Fox, M. and Poole, D., editors, *Proceedings of the 24th AAAI Conference on Artificial Intelligence*, pages 1413–1418.

Noessner, J., Niepert, M., and Stuckenschmidt, H. (2013). RockIt: Exploiting parallelism and symmetry for MAP inference in statistical relational models. In *Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence*.

Noy, N. F. (2004). Semantic integration: a survey of ontology-based approaches. *SIGMOD Record*, 33(4):65–70.

Noy, N. F. and Musen, M. A. (2000). PROMPT: Algorithm and tool for automated ontology merging and alignment. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 450–455.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Park, B.-H. and Kargupta, H. (2002). Distributed data mining: Algorithms, systems, and applications. In Ye, N., editor, *The Handbook of Data Mining*, pages 341–358. Lawrence Erlbaum Associates.

Pearl, J. (1988). *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Poole, D. (2003). First-order probabilistic inference. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, IJCAI'03, pages 985–991, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Poon, H. and Domingos, P. (2006). Sound and efficient inference with probabilistic and deterministic dependencies. In *Proceedings of the 21st National Conference on Artificial Intelligence*, volume 1 of *AAAI'06*, pages 458–463, Boston, Massachusetts.

Poon, H. and Domingos, P. (2007). Joint inference in information extraction. In *AAAI*, pages 913–918.

Poon, H. and Domingos, P. (2010). Unsupervised ontology induction from text. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 296–305. Association for Computational Linguistics.

Poon, H., Domingos, P., and Sumner, M. (2008). A general method for reducing the complexity of relational inference and its application to MCMC. In Fox, D. and Gomes, C. P., editors, *AAAI*, pages 1075–1080. AAAI Press.

Popov, B., Kiryakov, A., Ognyanoff, D., Manov, D., and Kirilov, A. (2004). KIM – a semantic platform for information extraction and retrieval. *Nat. Lang. Eng.*, 10(3-4):375–392.

Pujara, J., Miao, H., Getoor, L., and Cohen, W. (2013). Knowledge graph identification. In *International Semantic Web Conference (ISWC)*. Winner of Best Student Paper award.

Qin, H., Dou, D., and LePendu, P. (2007). Discovering executable semantic mappings between ontologies. In *Proceedings of International Conference on Ontologies, Databases and Applications of SEmantics (ODBASE 2007)*, pages 832–849.

Rahm, E. and Bernstein, P. A. (2001). A survey of approaches to automatic schema matching. *The VLDB Journal*, 10(4):334–350.

Richardson, M. and Domingos, P. (2006). Markov logic networks. *Machine Learning*, 62:107–136.

Riedel, S. (2008). Improving the accuracy and efficiency of MAP inference for Markov logic. In *Proceedings of the Proceedings of the 24th Conference on Uncertainty in Artificial Intelligence (UAI-08)*, pages 468–475.

Ritze, D., Meilicke, C., Svb-Zamazal, O., and Stuckenschmidt, H. (2008). A pattern-based ontology matching approach for detecting complex correspondences. In *Ontology Matching (OM-2009)*, volume 551.

Russell, S. J. and Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Pearson Education, 2 edition.

Shvaiko, P. and Euzenat, J. (2011). Ontology matching: State of the art and future challenges. *IEEE Transactions on Knowledge and Data Engineering*, PP(99).

Singla, P. and Domingos, P. (2006). Memory-efficient inference in relational domains. In *Proceedings of the 21st National Conference on Artificial Intelligence*, volume 1 of *AAAI'06*, pages 488–493, Boston, Massachusetts.

Sirin, E., Parsia, B., Grau, B. C., Kalyanpur, A., and Katz, Y. (2007). Pellet: A practical OWL-DL reasoner. *Web Semantics*, 5(2):51–53.

Todirascu, A., Romary, L., and Bekhouche, D. (2002). Vulcain - an ontology-based information extraction system. In *NLDB*, pages 64–75.

Venugopal, D. and Gogate, V. (2013). Giss: Combining gibbs sampling and samplesearch for inference in mixed probabilistic and deterministic graphical models. In desJardins, M. and Littman, M. L., editors, *AAAI*. AAAI Press.

Wang, J. and Domingos, P. (2008). Hybrid Markov logic networks. In *Proceedings of the 23rd National Conference on Artificial Intelligence*, volume 2 of *AAAI'08*, pages 1106–1111. AAAI Press.

Weld, D. S., Hoffmann, R., and Wu, F. (2008). Using Wikipedia to bootstrap open information extraction. *SIGMOD Record*, 37(4):62–68.

Wimalasuriya, D. C. and Dou, D. (2010). Ontology-Based Information Extraction: An Introduction and a Survey of Current Approaches. *Journal of Information Science*, 36(3):306–323.

Wu, F., Hoffmann, R., and Weld, D. S. (2008). Information extraction from wikipedia: moving down the long tail. In *KDD '08: Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 731–739, New York, NY, USA. ACM.

Yang, Q., Chen, Y., Xue, G.-R., Dai, W., and Yu, Y. (2009). Heterogeneous transfer learning for image clustering via the social web. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, volume 1 of *ACL '09*, pages 1–9, Stroudsburg, PA, USA. Association for Computational Linguistics.

Ye, J., Cheng, H., Zhu, Z., and Chen, M. (2013). Predicting positive and negative links in signed social networks by transfer learning. In *Proceedings of the 22Nd International Conference on World Wide Web*, WWW '13, pages 1477–1488.